

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest



COMPUTER AND AUTOMATION INSTITUTE,
HUNGARIAN ACADEMY OF SCIENCES

THE DESIGN OF COLOR, RASTER-SCAN GRAPHICAL DISPLAYS
FOR PROCESS CONTROL APPLICATIONS

Edgardo Felipe

Reports 55/1976

Responsible Publisher

T.Vámos

ISBN 963 311 027 0

Készült az Országos Műszaki Könyvtár és Dokumentációs
Központ házi sokszorosítójában
F.v.: Janoch Gyula

**THE DESIGN OF COLOR, RASTER-SCAN GRAPHICAL DISPLAYS
FOR PROCESS CONTROL APPLICATIONS**

Edgardo Felipe

**Director of Studies: József Hatvany
Cand. Tech. Sci.**

Budapest

1976

CONTENTS

	Page
INTRODUCTION	7
1. PROCESS CONTROL COLOR DISPLAY SYSTEMS	13
1.1 State-of-the-Art	14
1.2 Perspectives	20
2. CRITERIA FOR THE DESIGN OF OPTIMAL PROCESS CONTROL GRAPHIC DISPLAYS	23
2.1 Introduction	23
2.2 Definition of some terms used in this Chapter .	24
2.3 Total number of information bits required by a picture	25
2.4 Optimal shape of the cell	50
2.4.1 Condition for optimal subdivision in cells of any useful surface "S"	50
2.4.2 Hypothesis	51
2.4.3 Two criteria	51
2.4.3.1 Justification of Criterion 1	51
2.4.3.2 Justification of Criterion 2	65
2.5 Common characteristics of drawings in process control applications. Graphical Requirements ...	67
2.5.1 Alphanumeric characters	70
2.5.1.1 Letters	70
2.5.1.2 Numerals	71
2.5.1.3 Punctuation marks and editing symbols	72
2.5.2 Symbols oriented to the application	77
2.5.3 Graphics	79
2.5.3.1 Straight Lines	79
2.5.3.2 Intersections	81
2.5.3.3 Corners	87
2.5.3.4 Cutting of symbols	88
2.5.3.5 Types of straight lines	90

	Page
2.6 Length of Straight Lines	91
2.6.1 Saving of information bits in straight line definition	91
2.6.2 Saving obtained in repetition bits expressed in percent	110
2.6.3 Influence of partitioning on the total number of information bits "M" per instruction-word	111
2.6.4 Influence on the aspect ratio of the useful surface "S"	122
2.6.5 Possible saving of bits permitted by the accuracy of drawings	122
2.7 Selection of the number of dots per cell	127
3. THE DISPLAY AS A COMPONENT OF A PROCESS	137
3.1 Introduction	137
3.2 Definition of some terms used in this Chapter.	138
3.3 General characteristics observed in components of an industrial process	141
3.4 Basic information required about the process to be supervised and controlled	146
3.5 Most important coded information required to and from each component in the process	147
3.6 Manipulating System Generalities	148
3.6.1 Display Unit	154
3.6.1.1 Three design criteria for the display units	154
3.6.2 Computer Operating System	161
3.6.2.1 Tasks to be performed by the computer system with the in- formation from the process.....	161
3.6.2.1.1 Identification of the component	162
Types of Identifier- word formats	163

	Page
3.6.2.1.2 Component Vector.....	168
Data included in the	
Component Vector	170
3.6.3 Picture File	172
3.6.3.1 Virgin Program	173
3.6.3.2 Picture Modifying Program	175
3.6.3.3 Structure of the Picture File ...	177
3.6.3.4 Updating of Picture Files /PF/...	179
3.6.3.4.1 Sequence of opera-	
tions for picture	
updating	181
4. GRAPHICAL LANGUAGE	189
4.1 Introduction	189
4.2 Necessity of a Graphical Language	189
4.3 Creation of Pictures	197
4.3.1 Categories of the visual information to	
be handled by the display	199
4.3.2 Requirements for the Graphical Language .	200
4.4 Description of the Graphical Language	202
4.4.1 Conventions	202
4.4.2 Types of Statements	204
4.4.2.1 "Absolute Position" Statement ...	205
4.4.2.2 "Graphical" Statements. General-	
ities	206
4.4.2.2.1 "Text" Statement	211
4.4.2.2.2 "Element" Statement ...	213
4.4.2.2.3 "Call to Subroutine"	
Statement	214
4.4.2.3 "Move" Statement	216
4.4.2.4 "Declare Subroutine" Statement ..	220
4.4.2.5 "End of Subroutine" Statement ...	221
4.4.2.6 "End of Program" Statement	221
4.4.2.7 "End of File" Statement	222
4.4.3 Auxiliary Coded Information	222

4.4.3.1 Data, Local Variables and "Z" Values	222
4.4.3.1.1 Data	222
4.4.3.1.2 Local Variables	223
4.4.3.1.3 "Z" Values	226
4.4.3.2 "Mute" Code	226
4.4.4 Coding. Distribution of bits of statements	227
4.5 Complete example for a Pump Station	230
5. SIMULATION OF THE GENERAL CONCEPT PROPOSED FOR THE DISPLAY SYSTEM	241
5.1 Introduction	241
5.2 Objectives of the Simulation	241
5.3 Equipment used	242
5.4 Development of the task	242
5.5 Virgin Programs prepared	244
5.6 Experiences obtained with the Simulation	247
5.7 Conclusions for the present study	251
REFERENCES	259
Appendix 1	291
Appendix 2	297

INTRODUCTION

This investigation deals with some design considerations of color cathode-ray tube display systems for the supervision and control of industrial processes.

It relates to the general concept of an interactive cell-organized color raster-scan graphical display system oriented to process control applications. These have so far been designed "ad hoc" and decisions have been taken mostly by "rule of thumb".

Day by day, the use of graphical display systems for process control applications becomes more frequent in industrial environments, mainly in complex plants where reliable supervision and control by other means used so far is a relatively difficult task.

Up to the present, panel board type display systems have been used to monitor the real conditions of industrial plants. However, because of the complexity of the plants themselves, and the increase of information due to the expansion of the systems and to the use of computer controls, the panel board display systems have become larger and more complex. It has now become uncomfortable to monitor and control the plants using these means and often becomes complicated beyond practical operation.

As a result of this trend, the demand has emerged for a display system which is able to show a selected area of the plant in a relatively detailed form, and which can be connected easily to a computer system. Moreover it should facilitate reliable human communication with the process through the computing system, permitting adequate control of the part of the plant schematically displayed. This should increase the automatism and reliability of the process, so

that all these characteristics, conveniently handled, may increment the control efficiency and overall process productivity.

The main objective now is to present methodical design criteria based on the particular characteristics of the drawings required for the supervision, monitoring and control of industrial processes as well as on the requirements demanded by the application itself, in order to establish several general characteristics for a display system to be used for these tasks. As a result, not only the requirements to be taken into account in designing the graphical display units have been discussed, but also considering them as a part of the whole Supervision System, the general bases of a consistent Manipulating System have also been proposed. Finally, a special-purpose Graphical Language has been created which permits the construction of any type of drawing required by the application and renders all the necessary conditions to diminish the computer operative memory requirements. It permits also direct interactivity between the operator and the industrial process being supervised and controlled.

This study is composed of 5 Chapters. Chapter 1 discusses the state-of-the-art in several topics related with real-time systems oriented to the supervision and control of industrial processes and with the software created so far and used in this field. It includes also the results of the analysis made of the open literature mainly related to color raster-scan displays, and mentions some examples of display systems already implemented for the supervision and control of industrial plants. Unfortunately, no detailed descriptions of such specialized display systems and their operating systems has been found. From this arises the impossibility of effectuating a comparison and evaluation of the present study with respect to other works realized on these subjects. Finally, briefly the present perspectives of cell-organized

graphic displays used in this application are exposed.

Chapter 2 groups the theoretical part of this study. It includes as the most important topics dealt with, the deduction of a general expression for calculating the total number of information bits required to define any picture, based on the number and types of the different graphical elements which compose it. The optimal shape of the cells of a cell-organized graphical display is also determined in this part.

Additionally, the topological characteristics found to be common in the drawings required in process control applications have been enumerated. Based on these characteristics, the discussion of the graphical requirements is carried out in connection with the alphanumeric characters, the symbols oriented to this particular application and the necessary restricted graphics.

Finally, the demands related to the length of the straight lines to create the drawings required by the application have been analyzed in this Chapter. Based on the probability /or frequency/ of occurrence of straight lines of different length in drawings, a general expression which gives the corresponding limit probability of occurrence at which there is neither saving nor subutilized information bits for a given partition has been deduced. The analysis of this expression permitted us to realize a significant saving in information bits in the instruction-words, once the frequency of occurrence of straight lines with different length in drawings used in this application was determined. The discussion for the selection of the most commendable number of dots per cell has also been included.

Chapter 3 is of a descriptive character. It deals with some topics intimately related to the display units when they are considered as a component of a Process Supervision System.

Three design criteria for the display system are established and discussed. Additionally, the description of the Manipulating System proposed is included, as well as the detailed description of the different parts which compose the Picture File, its structure and the sequence of operations to be carried out for updating them.

Chapter 4 is related to the special-purpose Graphical Language created for the application. It includes the detailed description, limitations, syntax, coding and distribution of bits of the different types of statements proposed in order to fulfil the requirements previously established for the Graphical Language. The Chapter is complemented with a complete example illustrating the use of the statements and the auxiliary coded information proposed.

The Manipulating System proposed in this work can not be considered in any form as an isolated part of the whole Operating System handling the entire industrial process. It must be conveniently included in the system to permit it to carry out the man-machine and machine-plant communication for which it has been principally created. For this reason, the present study cannot be considered a fully comprehensive one, since many closely interrelated topics have not been considered, e.g. the facilities, methods, final objectives, etc. to carry out interactivity between the operator and the plant via the operating system, the analysis of the automatic methods to deal with emergency situations, the ergonomic studies related with the colored information shown on the display screen, etc. All these questions must be the subjects of future works in this field.

Chapter 5 deals with the Simulation carried out to show practically the realizability of the general concept proposed for the display system described and the validity of the criteria established in the present study. It includes the

results of 9 drawings prepared, all intimately related with the application dealt with and of actual and practical significance. Finally, the experiences obtained with the Simulation and the conclusions for the present study are also given in this Chapter.

The Appendices 1 and 2 are related to the complete example given at the end of Chapter 4 and consist in the lists of the non-updated and updated Virgin Program prepared by the Interpreter created to carry out the Simulation.

Reference [256] states with respect to the terms "picture" and "drawing":

... "We shall distinguish between drawings and pictures by the way in which each is described. Our basic premise is that pictures can represent "anything" and require quite general descriptive means, whereas drawings are a special class of pictures by virtue of more specialized descriptive means. Drawings can always be (re-)described as pictures, and pictures can almost always be (re-)described as drawings, albeit seldom efficiently. Thus, the name goes with the descriptive means, not with the artifact."

In spite of this, in the present study we have considered the terms "picture" and "drawing" as equivalent. Thus, we have not made any distinction between the two terms. They are considered as "the visual representation on any medium of any type of graphical or not graphical information".

During the development of the present investigation some other topics were analyzed which are not now included. The reasons are mainly the absence of space and their relatively smaller importance, e.g. the proposition of a practical Symbol-set for the application, the evaluation of the cursor and the light pen among other means to carry out interactivi-

ty with the process through the CRT's screens of color raster-scan display units, the discussion about frame interlacing, and others.

The present study has been carried out in the Division for the Mechanical Engineering Automation of the Computer and Automation Institute of the Hungarian Academy of Sciences.

Acknowledgement. I am very grateful to the many colleagues at the Computer and Automation Institute of the Hungarian Academy of Sciences who in one way or another have made possible the completion of the present work. Apart from my Director of Studies I wish to thank István Gallai, Albert Lábadi and Péter Darvas for the help given in the preparation of the Interpreter of the Graphical Language and in answering my many questions related to the TPA-70 Disk Operating System operating at the Institute. I am also very grateful to the diligent and courteous workers of the Library of the Institute for their services, to Luisa Molina who contributed to typing the rough draft of the manuscript and to Ms. J. Sántha who completed it and typed with great fondness the final version of the present study. Finally, my thanks are due to Ms. I. Cserhádi who typed the mathematical expressions, Ms. E. Zelenay who performed the photographic work and Ms. J. Schmidt who drew the drawings.

CHAPTER 1

PROCESS CONTROL COLOR DISPLAY SYSTEMS

The use of display systems oriented to process control applications is recent. There are not so far many written works in this field mainly due to the novelty of the subject and to the diverse requirements to be fulfilled due to the particular characteristics of the industrial processes. These characteristics are mainly related to the high reliability required, the immediate response demanded from the display system, the low cost-performance rate desired, the ergonomic requirements imposed and the present relative State-of-the-Art of the multiple branches of technology, in connection with the many diverse interrelated fields encountered in every industrial process to get an economic solution common to all them. This last characteristic, of course, is present continuously along the development of the technique. Finally there is an additional factor: the limited number of published papers related to the subject which is due mainly to significant economic reasons and primacy ambitions.

The importance of a reliable graphical display system in an industrial environment is unquestionable. It can be considered nowadays as an indispensable complement of highly automated industrial complexes for the supervision and control of the whole processes. Several difficulties arise, however, when a general solution compatible with the many kinds of actual processes and computing systems is desired.

As is common in the computing field, often the same equivalent final results can be attained either by means of software solutions or hardware implementations. Presently, the use of an intermediate solution /firmware/ has been con-

ceived, becoming more and more economically justifiable in many fields. With the development of computing theories, programming techniques, etc. together with the advances of technology, we are now nearing the common use of micro-processors in this field. In any case the selection of the best solution depends very greatly on the requirements imposed by the particular application field.

1.1 State-of-the-Art

Our main objective in this part, is to analyze the State-of-the-Art of those topics with which we shall have to deal.

Briefly, they refer to the cell organization of the display screen and the use of an object-oriented memory to store the visual information contained in the cells, to the requirements imposed as a consequence of the characteristics of drawings required by the application concerned, to a Manipulating System which permits us to handle the visual information, its updating and the interaction of the operator with the industrial process via the Operating System created for the whole system, and finally to the Special-purpose Graphical Language designed to create the pictures required by the application.

In the following pages a relatively small number of technical articles has been referred. The main reason is due to the approach followed through the study and the subjects concerned in it. This, in one or other form, is made evident with the analysis carried out in this Chapter of the present accessible literature nearly related and appearing at the end of this study. Because a relatively great number of papers has been revised, the list includes only the most recent ones and those appearing in the most important publications related with the field.

Early publications about works dealing with the control of

industrial processes by means of computers appeared at the beginning of the 1960s. The growing development of display technology began about the same time, becoming a common tool in engineering only at the end of the last decade.

However, the increasing use of the raster-scan TV principle /digital television/ paradoxically began a little later, in spite of the fact that the origin of TV systems dates back to several decades ago. First publications about digital television appeared in the first half of the 1960s dealing mainly with the display of alphanumeric characters [31, 35, 98, 167, 189, 251, 254].

Inconveniences of using an image-oriented memory in raster-scan displays have already been exposed in the literature [74, pp. 28; 194, pp. 4; 256, pp. 27; 277]. The need for a bit-map memory, particularly when a color implementation is desired, implies a prohibitive magnitude of memory capacity in order to get, at present, an economic solution. Nevertheless, several projects have been carried out following this alternative for attaining a general-purpose /color or black and white/ graphical display by means of the raster-scan principle either using costly core-memory or more cheap /from the point of view of the cost per bit/ disk drives for refresh purposes [116, 143, 148, 154, 214, 226, 263].

The use of an object-oriented memory presents advantages in many applications. This alternative originated from the use of a similar data structure in this kind of display as that of the random-scan graphical displays, but later it gave place to the so-called raster-scan cell-organized graphical display systems. Written works about this subject are really recent, the first articles appearing in the present decade [13, 128, 277, 295]. The advantages of the principle of the hypothetical subdivision in cells of a raster-scan CRT display's screen are indisputable. They are mainly reflected in the

reduction of memory requirements, the easy updating, the economic use of the color facility, the fulfilment of the requirements in quality of many applications, the overcoming of the inconveniences present when the bit-map solution is used, and others features all offering a considerable reduction in total cost, mainly in the case of multiterminal systems.

The use of the hypothetical cell subdivision of the display screen is promising, and is now increasingly popular. The difficulties arise when the same results, and therefore the same application fields are desired in this kind of display system as in a sophisticated random-scan display system. The trend is to create adequate algorithms to carry out the necessary scan conversion without requiring as so far, a considerable computer time for this task. Cell organization is, moreover, particularly recommended for matrix displays now in development which use other media to show visual information such as the plasma, liquid crystal and electroluminiscent display systems [78, 82, 91, 92, 118, 132, 259, 260]. The main question consists in finding more efficient and more powerful "lexical" facilities to express the general and particular context of visual information. In this form the visual information to be contained inside each independent cell could be somehow determined and after this conveniently coded in order to be handled by the computer system.

The recent reference [194] referring to raster-scan graphics in Computer-aided Design /CAD/, situates the present state of the principle of cell organization for storing and handling pictures to be shown in raster-scan CRT when used in CAD applications with respect to other methods. Chapter 2 of this study deals with the theoretical analyses and presents the results attained in connection with the use of this principle, when it is considered as a solution which permits us to save computer operative memory. The criteria stated therein are

based upon the particular characteristics of drawings required in process control applications.

In reference [74] we discussed and stated some general criteria for the selection of the best solution in connection with some basic questions related to the design of a process control graphical display system. It considered the State-of-the-Art at that moment and referred to the present publications in the open literature. For that reason, this study does not include topics dealt with therein, and other which are related to general questions in connection with display technology or with other types of display systems [1, 2, 7, 8, 10, 23, 26, 30, 31, 39, 40, 43, 49, 69, 72, 73, 74, 90, 100, 103, 106, 107, 125, 127, 134, 143, 148, 153, 154, 155, 157, 178, 179, 181, 189, 190, 195, 198, 204, 212, 218, 224, 226, 247, 252, 253, 255, 256, 266, 271, 277, 279, 283, 293]. Additionally, many other subjects not intimately related with process control display systems have not been discussed in this study. They are adequately analyzed and discussed in many other articles in the literature [11, 24, 33, 34, 41, 48, 54, 58, 67, 88, 89, 90, 93, 94, 105, 110, 124, 137, 140, 146, 147, 149, 150, 152, 157, 161, 166, 180, 197, 213, 257, 258, 265, 281, 282, 287, 291, 292].

Many works have been written also in connection with raster-scan displays, but oriented to other special applications: computerized page design [163], alphanumeric displays [1, 8, 35, 64, 98, 100, 127, 167, 214], low cost graphical terminals or systems [102, 103, 108, 158, 159, 172, 179, 201, 222, 263, 264], and others [68, 73, 90, 95, 101, 115, 129, 131, 143, 148, 154, 196, 198, 201, 205, 226, 254, 283, 293, 294].

Several graphic display systems oriented to process control applications have already been implemented [26, 36, 47, 96, 170, 185, 208, 219, 249, 254, 284, 288, 293, 296, 297, 299, 301, 303, 307, 308, 309]. Up to date, they are dealt with in

the literature in a very general fashion and with informative character. Many advertisements appearing in marketing brochures confirm this reality [307, 308, 309]. The theoretical analysis of related topics, studies about evaluations, comparisons among present implementations, standardization, economic analysis, etc. are seldom found yet [87, 296].

The excellent recent Survey of Gertler and Sedlak [87] justifies this evaluation when they expose that:

..."the present status of process control software included in the survey is mostly based upon a considerable amount of up-to-date written information, placed at the authors disposal by the courtesy of several leading vendors of process control systems" [pp.615].

The same article states:

"In process control application, the importance of the run-time efficiency of the programs in terms of Central Processing Unit time and core-space, is crucial. The application of high-level languages to process control programming is unquestionably advantageous from a couple of points of view, like the ease and quickness of program writing or transferability of programs. On the other hand, it inevitably introduces a certain degree of run-time inefficiency. This may be extremely critical in connection with some basic, very frequently executed functions, so these are advisable to program at lower levels, even if high-level techniques are used otherwise. Also, high-level programming may make on-line program testing and, especially, program modifications more complicated [pp.617].

Finally, in the Conclusion's part of that study it is stated that:

..."by now, there have been promising developments and some initial results, especially in the field of high-level general-purpose process control languages".

All this, besides justifying the considerations and points of view stated in Sect. 4.2 of our study, exposes clearly again the reduced number of publications about these subjects. Technical articles which have appeared in connection with programming languages created for process control applications, refer mainly to general-purpose process-control languages, which are sometimes real-time extensions of FORTRAN or based on other general algorithmic languages, or are more or less new ones [19, 63, 141, 176, 243].

In our present study the design of a special-purpose graphical language for creating pictures in a cell-organized color raster-scan display has been included. Its design is based also on the particular characteristics of drawings required for the supervision and control of industrial processes. Additionally to the factors stated above, a real necessity of such a language was present in the last stage of our study. The possibility of finding an already-designed graphical language which fulfills the same objectives was also present. The revision of many publications of recent years was undertaken, beginning of course with the surveys, rosters, technical reports, etc. that could in one or other form be relevant [27, 28, 32, 183, 232, 233, 234, 235, 236, 237, 241, 242, 274]. This patient search demonstrated to us no information about a graphical language which could fulfil our proposed objectives has been published. Nevertheless, papers about many other closely related special-purpose programming languages oriented to other applications,

graphical or not, have been found in the literature [21, 22, 29, 45, 46, 53, 63, 70, 80, 86, 95, 138, 142, 156, 160, 165, 168, 199, 217, 245, 262, 268, 270, 290]. This implied the designing of the special-purpose graphical language proposed in Chapter 4 of the present study. In principle, it consists of a macro-oriented language approach, this being also followed by other graphical languages at present. The necessity of its design and consequently its realization, corroborates the ideas formulated by Sammet in connection with the creation of special-purpose languages [231], it also being justified by our objectives and interest of giving completeness to the subjects dealt with.

We do not consider it an optimal solution, although it efficiently fulfills the requirements imposed by the application. The Graphical Language created has been proved by the Simulation carried out of the general concept of the display and Manipulating System proposed. The results and experiences obtained have been included in Chapter 5 of the present study.

1.2 Perspectives

The use of color graphical displays in process control display systems, presents many advantages as a result of the possibility of the color coding to indicate the very diverse possible contingencies, or simply to show the operator the actual state of the process being supervised.

The perspectives of the cell-organized principle proposed as an economic solution for a graphic display system which permits the continuous real-time supervision and control of industrial processes, are considerable. It permits the use of the raster-scan television principle with an appreciable saving in cost and a reduced requirement in memory capacity, mainly in color display implementations. The development of

more elaborate software for this principle is a topical demand of our days, also in order to get other objectives and to expand the application field of the raster-scan displays.

Other works have been written using this principle, some of them appearing in the bibliography. In this question, in connection with other applications, to make possible with the principle of the cell organization such facilities as clipping, windowing, scaling, automatic definition of continuous and variable shading and others which are already commonly encountered in costly sophisticated random-scan systems, it would create a new revolution in display technology, provided that the computing time demanded to carry them out does not become prohibitively long. This would include also implicitly the variability of the cell dimensions which should conduce to other interesting results.

One perspective based on the results obtained in the present study, is related to those new results arising after their future implementation in actual industrial plants. This promises particularly fruitful economic advantages in industrial complexes in which a great number of production units operate, e.g. in sugar-cane factories, in electric power plants, etc. Obviously, this will only be economically feasible in those cases where the available instrumentation is compatible with the considerable initial investment represented by the inclusion of highly automated means in an industrial environment.

CHAPTER 2

CRITERIA FOR THE DESIGN OF OPTIMAL PROCESS CONTROL GRAPHIC DISPLAYS

2.1 Introduction

This chapter deals with the theoretical analysis of some real demands for a low-cost, interactive graphical display oriented to process control applications.

Our objective has been to make the best use of the particular characteristics of drawings required for the supervision and control of industrial processes and for recording those data related to production, economic reports, statistical analysis, etc. commonly required by the application. For this, we have based our analysis on the topological characteristics of the drawings themselves in order to get the best solutions when such a kind of graphical display must be physically implemented. However, the deduction of a general expression which gives the total number of information bits required by a picture has been carried out on a general basis, that is, for two-dimensional pictures with arbitrary topology. The minimization of the expression deduced led us to establish some criteria in connection with the general principle according to which the display itself should operate.

The criteria established in this Chapter on the one hand support theoretically the bases of several implementations intuitively performed so far, and on the other hand they establish several conclusions which contribute to the optimal design of a graphical display to be used in process control applications.

2.2 Definition of some terms used in this Chapter

- Cell - Cell means each single part or portion of arbitrary geometric shape, in which a given Useful Surface "S" can be completely subdivided.
- Font - is the association of some graphical single unities belonging to a graphical entity, which have a more specific common property /i.e. the horizontal straight lines, the gothic alpha-numeric characters, the symbol set to be used in process control applications, etc./
- Graphical Element - is each one of the graphical unities belonging to a given font /i.e. the letter "C", the horizontal straight line of 7.6 cm of length, the symbol for a diode, etc./
- Graphical Entity - is the association of many graphical single unities which have an arbitrary but common general property /i.e. all the straight lines, all types of alphanumeric characters, etc./
- Useful Surface "S" - Useful Surface "S" means in this study the maximum surface of an arbitrary surface " S_0 ", which may be used to show visually the information required.

In all cases: $S \leq S_0$

Because of aesthetic reasons and in order to fit the shape of the Useful

Surface "S", to render manipulation easier, etc., it is always established that $S < S_0$

2.3 Total number of information bits required by a picture

The determination of the position of any point inside a plane surface with arbitrary shape and dimensions is defined uniquely with only two coordinates.

Let us consider a plane useful surface "S". Select a minimal number of points "N" inside it with which it shall be possible to define unambiguously any visual information located inside it. For convenience and to facilitate the mathematical analysis, we consider the points orderly arranged, that is, uniformly spaced forming a lattice of points ordered in parallel directions to the abscissa and ordinate axes of a Cartesian Rectangular Coordinate System.

Information Theory states that with a number "x" of information bits, it is possible to identify uniquely a maximum of 2^x elements using binary coding [83].

In general, for "x" elements are required a number of bits "x" which satisfy the expression:

$$x \leq 2^x$$

The special case

$$x = 2^x \quad (2.3.1)$$

takes place when the efficiency of the coding is maximum, being "x" the minimum number of information bits required to define uniquely all the elements "x".

The logarithm to the base 2 of Exp. 2.3.1 is:

$$\log_2 X = x \log_2 2$$

$$x = \log_2 X$$

Therefore, the minimal number of information bits "x" required to identify "X" elements with binary coding is equal to $\log_2 X$, if and only if $X \leq 2^x$.

In our case, for "N" points we have:

$$x = \log_2 N \quad \text{iff}^* \quad N \leq 2^x \quad (2.3.2)$$

Therefore, with an adequate coding of these "x" bits, it is possible to determine unambiguously the exact position of any of the "N" points composing the plane useful surface "S".

By hypothesis, in the simplest case, it is possible to show any kind of visual information inside "S" with only the existence or not of each point " N_1 " located into it. Because by convenience we supposed an ordered arranging of all the points, this implicitly presuppose the existence of all of them. Thus, in order to show simply the visual information, it is only required that the points be, for instance, blanked or unblanked.

Let us suppose that to show an arbitrary information on an useful surface "S" having "N" points, are sufficient "N'" points, i.e. $N' \leq N$.

Since to indicate the blanking or unblanking of any point is required only one information bit, then for "N'" points

* In this study, the term "iff" is used to designate the conditional statement "if and only if" required many times in our mathematical formulations.

are required "N'" information bits.

Since each one of these points requires to be positioned in order to settle its corresponding information, then for "N'" points are required, taking into account the Exp. 2.3.2:

$$i_1 = N' \log_2 N \quad \text{information bits} \quad (2.3.3)$$

Because the information about the state of each point of every picture created each time, normally requires to be stored in some form somewhere for a later specific purpose /refreshing, picture retrieval, etc./, then

$$i_2 = N \quad (2.3.4)$$

information bits are additionally required, from which "N'" are unblanked and "N-N'" remain blanked.

As result of this, to define, to settle and to store the information of "N'" points of an useful surface "S" with "N" points, adding Exp. 2.3.3 and Exp. 2.3.4 it is seen that totally:

$$\begin{aligned} I &= i_1 + i_2 \\ I &= N' \log_2 N + N \end{aligned} \quad (2.3.5)$$

information bits are required.

Since always $N' \leq N$, then:

$$N' = k_0 N \quad (2.3.6)$$

where $0 \leq k_0 \leq 1$.

The value of " k_0 " gives the ratio of the total "N" points, utilized for showing a particular visual information on "S", that is, a measure of the density of the picture being represented.

Substituting Exp. 2.3.6 in Exp. 2.3.5:

$$I = k_0 N \log_2 N + N \quad (2.3.7)$$

$$I = N(k_0 \log_2 N + 1) \quad (2.3.8)$$

Exp. 2.3.8 gives the total number of information bits "I" required by a monochromic raster-scan graphic display with bit-map memory to define, to settle and to store the information of a picture utilizing a fraction " k_0 " of the total "N" points of the useful surface "S" on the screen.

Then, the total number of information bits required in this case, depends on the amount of addressable points "N" and on the density of the information shown in the picture.

In this kind of display when they are used in other applications than alphanumeric display, the determination of those points to be blanked or unblanked to show any visual information in a comprehensible form, requires many calculations. These, of course, are done by the minicomputer which handles the display, but usually it is a very computer time-consuming task. The bigger the complexity of the picture, the larger the computer time required to process the information required. That is a consequence of the principle of work of this type of displays, because the beam path is fixed and only beam intensity can be varied.

A random-scan graphic display's principle of work is very different. In this kind of display, by means of a relatively complex hardware, the necessary conditions are created in order to produce and display on the screen the elements of the different types of graphic entities included. By means of character generators, vector generators, etc. it is possible to display in a straightforward form each graphic element, one after each other. Then, in principle, the display considers only some types of graphic entities with

which it is possible to produce all the necessary graphic elements to compose the pictures. Generally, these hardware facilities are costly to implement, because of the requirements imposed. Commonly, these displays make use of both absolute and relative positioning facilities to settle in place the different graphic elements on the screen.

To deduce a general expression which will permit us to calculate how many information bits are required to define the visual information included into an arbitrary graphical picture displayed on the screen of any type of display, we analyze the most general case, that is, an arbitrary graphical picture composed of an arbitrary number of graphical elements, which belong to different fonts of different types of graphical entities.

The definition of these terms given in Sect. 2.2 is represented graphically in Fig. 2.1

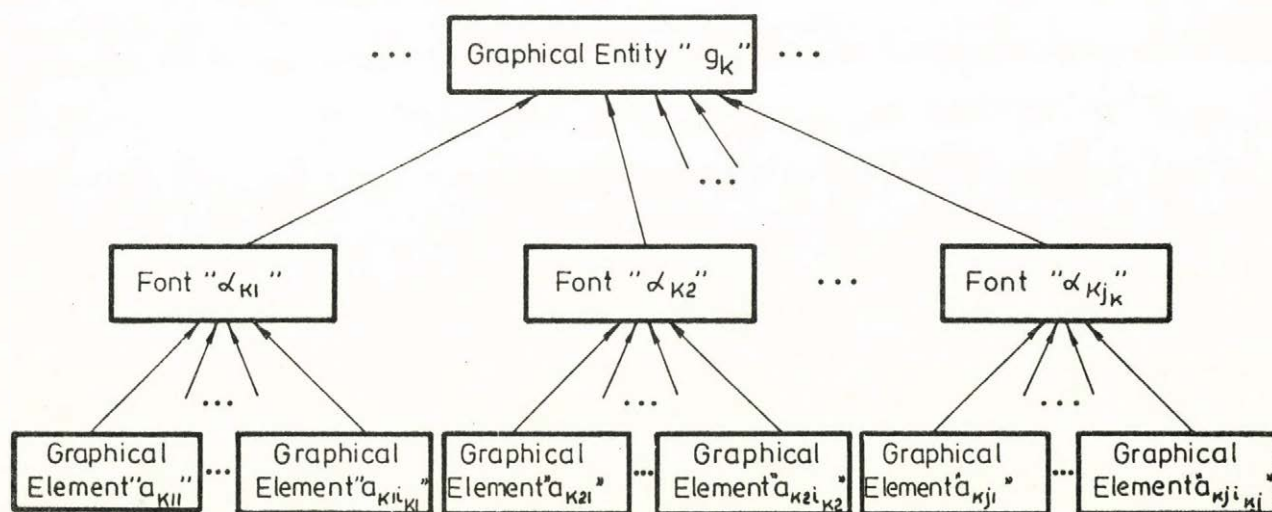


Fig. 2.1

For example, the graphical entity " g_k " could associate all the alphabets; the font " α_{k1} " could be the association of

the letters of English alphabet, the font " α_{k2} " could be the association of the letters of the Cyrillic alphabet, etc.; the graphic element " a_{k11} " of the font " α_{k1} " could be the letter "A", the second one could be the letter "B" and so on; the graphic element " a_{k21} " of the font " α_{k2} " could be the letter "B", the second one could be the letter "И", and so on. Normally, as a general rule, the graphical elements follow a logic order to facilitate their identification.

The following analysis will be done considering only the existence of graphical entities, fonts and graphical elements, but it does not exclude, in any other display system, the possible existence of sub-graphical entities and sub-fonts. In that case the final expression obtained must be further generalized.

With these concepts in mind, suppose that the most complex picture " M_0 " requires " k " different graphical entities " g ", that is:

$$g_1, g_2, g_3, \dots, g_{k-1}, g_k$$

where " k " is an integral number in the range $1 \leq k < \infty$

Using binary coding to identify each graphical entity " g_k " among them, a minimum number of information bits given by:

$$\begin{aligned} y &= \log_2 k & 1 \leq k < \infty & \quad (2.3.9) \\ \text{iff } k &\leq 2^y & k: \text{integral} & \end{aligned}$$

are required.

Suppose now that each type of graphical entity " g " has " j " different fonts. Let " α_{kj_k} " be the font " j_k " of the graphical entity " k ", thus:

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \dots, \alpha_{1j_1} \quad 1 \leq j < \infty, j: \text{integral}$$

are the fonts of the graphical entity " g_1 ",

$$\alpha_{21}, \alpha_{22}, \alpha_{23}, \dots, \alpha_{2j_2} \quad 1 \leq j < \infty$$

are the fonts of the graphical entity " g_2 " , and so on.

In general: $\alpha_{1j_1} \neq \alpha_{2j_2} \neq \dots \neq \alpha_{kj_k}$ that is, we suppose that each graphical entity has a different number of fonts.

Suppose also that each font " α_{kj_k} " has " i " different graphical elements. Let " a_{kji} " be the graphical element " i " of the font " j " of the graphical entity " k ", thus:

$$a_{111}, a_{112}, \dots, a_{11i_{11}} \quad 1 \leq i < \infty, i: \text{integral}$$

are the graphical elements of the font " α_{11} ",

$$a_{121}, a_{122}, \dots, a_{12i_{12}}$$

are the graphical elements of the font " α_{12} " , and so on.

In general:

$$a_{11i_{11}} \neq a_{12i_{12}} \neq \dots \neq a_{21i_{21}} \neq a_{22i_{22}} \neq \dots \neq a_{kj_i_{kj}}$$

that is, we suppose that each font has a different number of graphical elements.

Moreover, let " j_k " be the total number of fonts of the graphical entity " g_k " and " a_{kj_k} " the total number of graphical elements of the font " α_{kj_k} " .

With these established, we can begin the deduction of the general expression.

To identify the font " α_{1j_1} " uniquely from all the other fonts of the graphical entity " g_1 ", a minimum of:

$$z_1 = \log_2 j_1 \quad \text{iff } j_1 \leq 2^{z_1}$$

information bits are required, where " j_1 " is the total number of fonts of the graphical entity " g_1 ".

To identify the font " α_{2j_2} " uniquely from all the other fonts of the graphical entity " g_2 ", a minimum of

$$z_2 = \log_2 j_2 \quad \text{iff } j_2 \leq 2^{z_2}$$

information bits are required, where " j_2 " is the total number of fonts of the graphical entity " g_2 ".

Then, to identify the font " α_{kj_k} " uniquely from all the other fonts of the graphical entity " g_k ", a minimum of:

$$z_k = \log_2 j_k \quad \text{iff } j_k \leq 2^{z_k}$$

information bits are required, where " j_k " is the total number of fonts of the graphical entity " g_k ".

Since the identification of different graphical elements among them is done normally one by one, we can consider the use of only one register to decode all the coded information. In this case the register, commonly called buffer register, must have enough place to lodge the longest code. For that reason from the different values obtained above for " z ", i.e. z_1, z_2, \dots, z_k , we select the larger one. Let " J " be the number of fonts belonging to the graphical entity /or graphical entities/ which have the larger number of fonts. Thus, the expression:

$$z = \log_2 J \quad \text{iff } J \leq 2^z \quad (2.3.10)$$

gives us the number of information bits which permits to identify uniquely each time any one font from all other fonts belonging to any graphical entity from the "k" possibles.

On the other hand, to identify the graphical element " $a_{11i_{11}}$ " uniquely from all the other graphical elements of the font " α_{11} ", a minimum of:

$$x_{11} = \log_2 a_{11} \quad \text{iff} \quad a_{11} \leq 2^{x_{11}}$$

information bits are required, where " a_{11} " is the total number of graphical elements in the font " α_{11} ".

To identify the graphical element " $a_{12i_{12}}$ " uniquely from all other graphical elements of the font " α_{12} ", a minimum of:

$$x_{12} = \log_2 a_{12} \quad \text{iff} \quad a_{12} \leq 2^{x_{12}}$$

information bits are required, where " a_{12} " is the total number of graphical elements in the font " α_{12} ".

Then, to identify the graphical element " $a_{1j_1i_{1j_1}}$ " uniquely from all the other graphical elements of the font " α_{1j_1} ", a minimum of:

$$x_{1j_1} = \log_2 a_{1j_1} \quad \text{iff} \quad a_{1j_1} \leq 2^{x_{1j_1}}$$

information bits are required, where " a_{1j_1} " is the total number of graphical elements in the font " α_{1j_1} ".

Similarly, this occurs to identify the graphical element " $a_{kj_ki_{kj_k}}$ " uniquely from all the graphical elements of the font " α_{kj_k} " ($1 \leq k < \infty$).

For the same reason than above, the larger value of "x"

from the possible values $x_{11}, x_{12}, \dots, x_{ij}, \dots, x_{kl}, \dots, x_{kj}$ is selected. Let "I" be the number of graphical elements belonging to the font /or fonts/ which have the larger number of graphical elements. Thus the expression

$$x = \log_2 I \quad (2.3.11)$$

gives us the number of information bits required to identify uniquely any one graphical element from the remainder belonging to any font from the possible $\sum_{t=1}^k j_t$ fonts.

Adding Exp. 2.3.9, Exp. 2.3.10 and Exp. 2.3.11:

$$Z' = \log_2 k + \log_2 J + \log_2 I$$

$$Z' = \log_2 (kJI) \quad (2.3.12)$$

Exp. 2.3.12 gives the number of information bits required to identify uniquely any graphical element from all the graphical elements possible to be displayed.

Now, from Exp. 2.3.12 we can not conclude that the best use of the buffer register has been achieved. Moreover, it is not possible to affirm surely that the total number of information bits "Z'" given by Exp. 2.3.12 is the minimal one.

By hypothesis, we have that " a_{kj_k} " is the total number of graphical elements of the font " a_{kj_k} ". Moreover, " j_k " is the total number of fonts of the graphical entity " g_k ".

With this in mind, we can write that the total number of graphical elements belonging to the font " a_{11} " is given by " a_{11} "; the total number of graphical elements belonging to the font " a_{12} " is " a_{12} ", and so on. Thus, the total number of graphical elements in all the fonts " j_1 " of the graphical entity " g_1 " is given by the expression:

$$x_1 = a_{11} + a_{12} + \dots + a_{1j_1}$$

$$x_1 = \sum_{t=1}^{j_1} a_{1t}$$

Similarly for the graphical entity "g₂" , the total number of graphical elements in all their fonts "j₂" is given by the expression:

$$x_2 = \sum_{t=1}^{j_2} a_{2t}$$

Generalizing, for the graphical entity "g_k" we have:

$$x_k = \sum_{t=1}^{j_k} a_{kt}$$

Then, the absolute total number of graphical elements belonging to all the fonts of all the graphical entities is:

$$x = x_1 + x_2 + \dots + x_k \quad (2.3.13)$$

Substituting expressions above in Exp. 2.3.13:

$$x = \sum_{t=1}^{j_1} a_{1t} + \sum_{t=1}^{j_2} a_{2t} + \dots + \sum_{t=1}^{j_k} a_{kt}$$

$$x = \sum_{s=1}^k \left(\sum_{t=1}^{j_s} a_{st} \right) \quad (2.3.14)$$

Therefore, Exp. 2.3.14 gives the total number of graphical elements available for the system.

Now, if "x" can be expressed exactly as a power "Z'" of 2 /if it is not, we can do it/, we can write:

$$2^{Z'} = \sum_{s=1}^k \left(\sum_{t=1}^{j_s} a_{st} \right) \quad (2.3.15)$$

If Exp. 2.3.15 is fulfilled, then the value "Z'" gives the minimum number of information bits required to identify uniquely all the elements "x" with maximal efficiency in coding. The value for "Z'" will then be:

$$Z' = \log_2 \left[\sum_{s=1}^k \left(\sum_{t=1}^{j_s} a_{st} \right) \right]$$

Expanding the inner sum in expression above:

$$Z' = \log_2 \left[\sum_{s=1}^k (a_{s1} + a_{s2} + \dots + a_{sj_s}) \right] \quad (2.3.16)$$

If every font has the same number of graphical elements, we can assure that the minimum number of bits in the buffer register assigned to decode the graphical elements belonging to any font are used totally with all the fonts, because the number of bits required is always the same for each one.

In that case:

$$a_{s1} = a_{s2} = \dots = a_{sj_s} = a$$

Substituting this in Exp. 2.3.16:

$$Z' = \log_2 \left(\sum_{s=1}^k j_s a \right)$$

Expanding this expression:

$$Z' = \log_2 [a(j_1 + j_2 + \dots + j_k)] \quad (2.3.17)$$

Similarly as above, if every graphical entity has the same number of fonts, we can also assure that the minimum number of bits in the buffer register assigned to decode the fonts belonging to any graphical entity are used totally with all the graphical entities, because the number of bits required is always the same for each one.

In that case:

$$j_1 = j_2 = \dots = j_k = j$$

Substituting this in Exp. 2.3.17:

$$Z' = \log_2 kja \quad (2.3.18)$$

Comparing Exp. 2.3.18 and Exp. 2.3.12 it is possible to see that they are similar, because if every graphical entity is composed of the same number of fonts and every font is composed of the same number of graphical elements, then the larger graphical entity and larger font can be considered to be whichever from those possible ones. However, now certainly the minimal number of bits are used.

Thus, the following Criterion can be established:

Criterion: In a graphical system with "k" different graphical entities " g_k ", having each " j_k " different fonts " α_{kj} " each and having " a_{kjk} " different graphical elements each, the minimal number of information bits required to identify /or define/ any graphical element from those possible, is obtained when each type of graphical entity has the same number of fonts and each font has in his turn the same number of graphical elements.

The minimal total number of information bits required is given by the expression:

$$Z' = \log_2 kja = \log_2 kJI \quad \text{iff} \quad 2^{Z'} = kja = kJI$$

Thus, the best use is given now to the buffer register used to decode each graphical element and for a given number of them, the efficiency of the coding will be maximum.

This result coincides with those stated by Information Theory related to fixed-length codes [83].

In the successive parts we shall use the notation k, J, I for the graphical entities, for the number of fonts which each graphical entity has and for the number of graphical elements which each font has, respectively.

Then with this stated, if the arbitrary picture " M_0 " has " B " graphical elements of any kind, then to compose it are required a minimum of:

$$Z'' = B(\log_2 k + \log_2 J + \log_2 I) \quad (2.3.19)$$

information bits.

Now then, for " N " addressable points:

$$Y_N = \log_2 N \quad \text{iff} \quad N \leq 2^{Y_N}$$

information bits are required additionally to define the position information.

If from the total number " B " of graphical elements displayed of all the fonts, only " b " graphical elements require explicit absolute position information because " $B-b$ " elements can be settled in place by means of relative positioning,

then there are required:

$$Y'_N = b \log_2 N \quad (2.3.20)$$

additional information bits for position.

Adding Exp. 2.3.19 and 2.3.20, we see that the total number of information bits "Z" required to compose an arbitrary graphical picture "M₀" is given by the expression:

$$Z = B(\log_2 k + \log_2 J + \log_2 I) + b \log_2 N \quad (2.3.21)$$

From Exp. 2.3.21 we can conclude that the total number of information bits "Z" depends on:

- the total number of different graphical entity types "k"
- the largest number of fonts "J" in a graphical entity
- the largest number of graphical elements "I" in a font
- the total number of addressable points "N" in the useful surface "S"
- the total number "B" of graphical elements displayed to compose the picture "M₀"
- the total number "b" of graphical elements which require absolute positioning.

The first term of Exp. 2.3.21, as a whole, can be considered as the necessary and sufficient number of bits required for defining and of course, for identifying and storing the total visual information being displayed. The second one represents the number of bits required for the absolute position of those graphical elements which require it. This position information, in general, must also be stored in some way.

With the former in mind, Exp. 2.3.21 can be used now, for instance, to determine the total number of bits required by a raster-scan display with bit-map memory.

In this type of display the number of bits required to store any picture is always the same, given by the total number of addressable points "N" on the screen. It does not depend on the picture displayed. Then, the first term of Exp. 2.3.21 in this case can be considered to be equal to "N".

On the other hand, by the principle of work of this kind of display, the visual information is formed unblanking independently point by point. Those which require to be unblanked, are defined by their coordinates, that is, using only position information. If the number of points unblanked is "N'", we can represent it as a fraction " k_0 " of the total number of points "N". Then:

$$N' = k_0 N, \quad 0 \leq k_0 \leq 1$$

Since the totality of points "N'" containing information requires positioning, then the term $k_0 N \log_2 N$ will be the value of the second term in Exp. 2.3.21.

Adding both terms we obtain the total number of bits required, i.e.:

$$Z = N + k_0 N \log_2 N$$

$$Z = N(k_0 \log_2 N + 1)$$

which is equal to the Exp. 2.3.8 previously deduced for this type of display.

To illustrate this in another way and in order to prove the validity of Exp. 2.3.21, the values acquired for the variables in this expression for this kind of display are now determined. In this case:

$k = 1$ since there is only one graphical entity: points

- J=1 since the font in all cases is only one: the selected point
- I=2 since each point can be unblanked or mantained blanked
- B=N since it is necessary to dispose of the totality of the addressable points to compose the pictures
- N=N since it is the number of addressable points
- $b=k_0 N$ since it represents the number of points which require absolute positioning when a picture is created.

Then, substituting these values in Exp. 2.3.21 we have:

$$Z = N(\log_2 1 + \log_2 1 + \log_2 2) + k_0 N \log_2 N$$

$$Z = N + k_0 N \log_2 N$$

$$Z = N(k_0 \log_2 N + 1)$$

which is again equal to the Exp. 2.3.8.

Now then, suppose that we have a finite useful surface "S" which can be completely defined with only one point i.e. $N=1$ and suppose also that by means of some method we can display "M" different pictures as complex as possible in a comprehensible form ($1 \leq M < \infty$). By using Exp. 2.3.21, we will compute the total number of information bits required to display "i" different or equal pictures. In this case:

- $k=1$ since there is only one graphical entity: pictures
- J=1 since there is only one font: the pictures to be displayed
- I=M since there are "M" different pictures
- B=i since the number of pictures displayed is "i"
- $b=i$ since all the pictures require absolute positioning, although always the same
- $N=1$ since there is only one addressable point

Substituting these values in Exp. 2.3.21 we have:

$$Z = i(\log_2 1 + \log_2 1 + \log_2 M) + i \log_2 1$$

$$Z = i \log_2 M$$

Although this case seems trivial, it is not so. This situation takes place with the slide projecting system which constitutes another type of display. They do not need different position information for each graphic element, because all them are displayed in only one "point": the screen. Then, the quantity represented by "Z" ($Z = i \log_2 M$) will be effectively the total number of information bits required to select "i" equal or different slides among "M" possibilities. This clearly also represents the minimal total number of bits required to store the visual information contained in "i" slides considered as a whole.

This result also proves the validity of Exp. 2.3.21.

Exp. 2.3.21 may also be written as:

$$Z = B \log_2 kJI + b \log_2 N$$

Transforming this expression we have:

$$Z = \log_2 (kJI)^B + \log_2 N^b$$

$$Z = \log_2 (kJI)^B N^b \quad (2.3.22)$$

From Exp. 2.3.22, by Information Theory, the total number of information bits "Z" required to define the information in an arbitrary graphical picture "M₀" will be a minimum and the efficiency of coding will be a maximum, if and only

if the expression:

$$2^Z = (kJI)^B N^b$$

is fulfilled.

We observe from Exp. 2.3.21 that the dependence of "Z" on the total number of graphical entities "k", on the largest number of fonts into a graphical entity "J", on the largest number of graphical elements "I" into a font and on the number of addressable points "N", is similar in weight in all cases. Additionally, we also observe that the influence on "Z" of "k", "J" and "I" in a practical design will be larger than that of the value "N", because always $B \gg b$.

Then the main goal for minimizing the value "Z" consists in reducing as far as possible the values of "k", "J" and "I", and finally to reduce as much as possible the value of "N".

Two alternatives can be possible:

1. To select powerful graphical entities which permit us to get the higher number of graphical elements without the necessity of a high number of fonts.
2. To select a minimum number of fonts all of them belonging to only one graphical entity and with a minimum number of graphical elements each.

The selection between these two alternatives depends, in the first place, on the application in which the display will be used.

An example of a display which uses the first alternative is the sophisticated graphical display; the common alpha-numerical display is an example which uses the second alternative.

The sophisticated graphical display makes use, for example, of the so-called vector generators. These are powerful means created by hardware with which the tracing of straight lines of different lengths and slopes is possible. In this case, the graphical entity could be considered as that which associates all the vectors conceived. As sub-graphical entity can be considered the association of all the straight lines with arbitrary lengths and slopes which can be traced on the useful surface of the screen. For tracing any given straight line, all that is required is to select the values for the corresponding parameters which define uniquely the particular straight line. To give validity to Exp. 2.3.21, if each one of the required parameters is considered as one font, the total number of graphical elements in this case are all the different straight lines which can be created with the combinations of all the values /or ranges/ of each one of the parameters specified.

If we analyze now the possibilities of the circle generator, we can make similar considerations. We can consider the straight line as a special case of a circle, that is, as a segment of the circle with a radius of infinite length. In this case, the vector generator could be considered as a font /or as a sub-graphical entity/ of the graphical entity which associates all the circles, being another font, for example, that which would permit us to trace all the circles with radius less than certain value. In this case, the parameters to identify each graphical element, must be considered now as sub-fonts. Hence, we see that the Exp. 2.3.21 can always be applicable depending on the considerations stated.

Finally, we know that the sophisticated graphical displays use generally random-scanning to produce the pictures on the screen. This scanning method permits the successive use of the different types of generators /used for the generation

of vectors, characters, circles, etc./ one after each other, depending only on the program prepared.

Returning to Exp. 2.3.21, it is seen that in all cases we obtain positive and integral values for "Z". This means that the only way to diminish the number of information bits per picture is using either in the system:

- a small total number of graphical entities "k"
- a small total number of fonts "J" per graphical entity
- a small total number of graphical elements "I" per font
- a small number of graphical elements "b" requiring absolute position information
- and a minimal number of addressable points "N" on the useful surface.

Obviously, no influence can be exerted over the value "B" because it depends on the specific picture displayed.

We shall analyze now some alternatives in order to minimize the Exp. 2.3.21. First of all, we shall analyze the term " $\log_2 N$ ".

This term gives the total number of information bits required to position absolutely all the graphical elements "b" which require it, belonging to any font, on whichever of the possible addressable points "N" of the useful surface "S".

In the most general case, every graphical element composing a picture requires position information to be settled in place on the useful surface "S". However, it does not have any visual information itself, though it is necessary for the composition of the picture in a comprehensible form to the observer or operator. As its frequency of utilization

increases with the complexity of the picture, some solutions have been found to diminish the total number of information bits used for positioning operation. The most important one is the facility of relative positioning in almost all graphical display, thus saving position instructions since the different graphical elements are settled like if they were "connected" or "joined" together with each other. Nevertheless, the use of absolute positioning is indispensable to position physically independent parts of the whole picture, or at least, at the beginning of the picture's tracing.

It is of great importance not only to minimize the number of times the position information must be used, but to see that the number of information bits required for positioning be a minimum. To give higher flexibility to present displays, the number of addressable points "N" on the screen has been commonly raised as high as possible, being limited mainly by the cost. This is the case of the costly sophisticated graphical displays.

One of the main goals in this study is to meet the conditions for a graphical display of wide applicability, but with low cost. For this reason, we analyze the possibilities of using a lesser number of addressable points for positioning operation of graphical elements without considerable loss of flexibility.

Let

$$X = \log_2 N \quad N \geq 1 ; \quad X \geq 0 \quad (2.3.23)$$

be a function where "N" is the total number of addressable points on the useful surface "S" and "X" the minimum number of information bits required to define uniquely any of them /See Exp. 2.3.2/.

The general mathematical expression of the logarithmic function is [311]:

$$x = \log_b \frac{Y}{a} \quad (2.3.24)$$

Exp. 2.3.24, with similar conditions to those of the Exp. 2.3.23, that is:

$$\frac{Y}{a} \geq 1 ; \quad x \geq 0$$

has a relative minimum for "x" in the value $y=a$ /Fig. 2.2/.

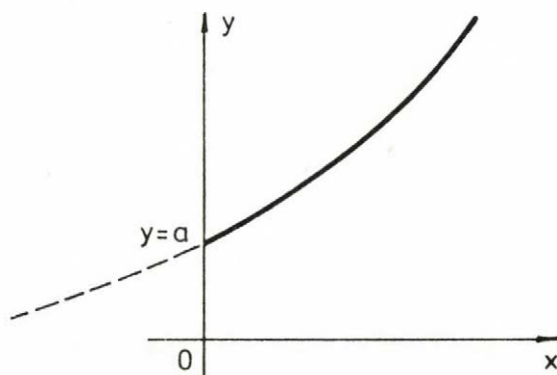


Fig. 2.2

If Exp. 2.3.23 is written in a similar form, we have:

$$X = \log_2 \frac{N}{1} \quad N \geq 1 ; \quad X \geq 0$$

Hence, comparing this expression with Exp. 2.3.24, it is seen that it has a relative minimum in $N=1$ in the range specified. This is the case of the slide projecting display system analyzed above; a position information lesser than this is not possible.

If we write now the Exp. 2.3.23 in the form:

$$X = \log_2 h \frac{N}{h} \quad N \geq h \geq 1 ; \quad X \geq 0 \quad (2.3.25)$$

it remains the same.

Exp. 2.3.25 can also be written as:

$$X = \log_2 h + \log_2 \frac{N}{h} \quad (2.3.26)$$

In this expression, the term $\log_2 \frac{N}{h}$, if $\frac{N}{h} \geq 1$, has a minimum in $N=h$ by similar considerations than formerly.

Exp. 2.3.26 can be interpreted as a solution for our purpose, if the useful surface "S" having a total number of addressable points "N" is completely subdivided in " $\frac{N}{h}$ " equal portions each one having "h" points. Thus, " $\frac{N}{h}$ " is now the arbitrary number of addressable points /from the total number "N"/ for each equal portion " $\frac{N}{h}$ " obtained as product of the subdivision. For this consideration, of course, we have considered an uniformly ordered arrangement of all the points on the surface.

Now if we consider the quantity of points "h" as the necessary and sufficient amount of points to represent a unity of visual information, then the term $\log_2 \frac{N}{h}$ can be considered now as the number of information bits required to position each one of these unities of information on the useful surface "S".

The term

$$X = \log_2 h \quad N \geq h \geq 1 ; \quad X \geq 0$$

gives us the number of information bits saved with this artifice.

Then, we can conclude that the number of information bits

required for positioning has been reduced in a quantity equal to $\log_2 h$, i.e.

$$X' = X - \log_2 h = \log_2 \frac{N}{h}$$

represents the number of bits required now to position each unity of information created.

The result of this analysis is no more than the solution given to the well-known alphanumerical display, where the unities of information consist in a specific and well-known limited number of alphabet letters, digits and punctuation marks from a given font.

In this study, based on the above, we propose a similar solution for an interactive graphical display oriented to process control application.

As a result of this, we can consider now as the only one type of graphical entity, those portions in which the useful surface have been subdivided. Thus, in our analysis we have now:

$$k = 1$$

which is the minimal possible value for this variable.

In order to minimize the other parts of this term, that is the value "b", three solutions can be proposed:

- to make a great use of relative positioning in creating the pictures,
- to make a great use of the repetition facility to represent the graphical elements,
- to make a great use of graphical subroutines, i.e. to use macroinstructions to represent those part in the picture which have the same topology.

We shall consider these three alternatives as new criteria

2.4.2 Hypothesis

In order to get practical results, the following hypothesis is stated for further analysis:

- All the cells must have the same area.

Then: $s_1 = s_2 = \dots = s_n = s$

Therefore, from Exp. 2.4.1:

$$ns = S \quad (2.4.2)$$

where: S is the total area of the useful surface
 n is the total number of cells within the surface
 s is the area of a single cell.

2.4.3 Two criteria

To determine the optimal shape of the cell the following two criteria are established:

Criterion 1. The optimal geometric shape of the cell, must satisfy the Exp. 2.4.2 necessarily.

Criterion 2. The cell of optimal geometric shape must not require change in orientation to fulfil the Exp. 2.4.2.

2.4.3.1 Justification of Criterion 1

The justification of first criterion will consist in two parts, one considering the most internal cells of the useful surface "S" and the other one considering the most external cells, that is, those which are located on the border constituting the edges of the useful surface "S".

1. Considering the most internal cells

Let "A" be a cell of arbitrary shape /Fig. 2.3/ with "n" sides l_1, l_2, \dots, l_n and area "s", which belongs to the useful surface "S". Let $\theta_1, \theta_2, \dots, \theta_n$ be the internal angles and $\alpha_1, \alpha_2, \dots, \alpha_n$ be the corresponding supplementary angles.

In this analysis, the following two hypotheses are stated:

Hypothesis 1. The cell must be convex.

Hypothesis 2. The area " A_A " of the cell "A" is considered much less than the area " A_S " of the useful surface "S", i.e. $A_A \ll A_S$.

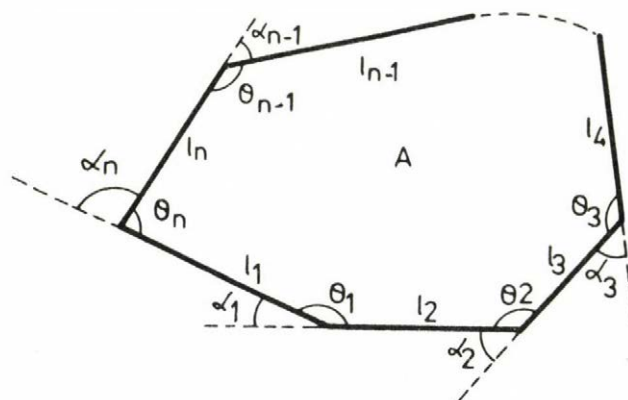


Fig. 2.3

In Fig. 2.3, by elemental geometry:

$$\begin{aligned} \alpha_1 &= \Pi - \theta_1 \\ \alpha_2 &= \Pi - \theta_2 \\ &\vdots \\ \alpha_n &= \Pi - \theta_n \end{aligned} \tag{2.4.3}$$

Since the sum of all external angles of any closed convex geometric figure is always 2π , then:

$$\alpha_1 + \alpha_2 + \dots + \alpha_n = 2\pi \quad (2.4.4)$$

Substituting Exp. 2.4.3 in Exp. 2.4.4

$$\begin{aligned} \pi - \theta_1 + \pi - \theta_2 + \dots + \pi - \theta_n &= 2\pi \\ n\pi - (\theta_1 + \theta_2 + \dots + \theta_n) &= 2\pi \end{aligned} \quad (2.4.5)$$

Exp. 2.4.5 is valid for all convex cells.

Now an arbitrary vertex "V" of the cell "A" is analyzed /Fig. 2.4/.

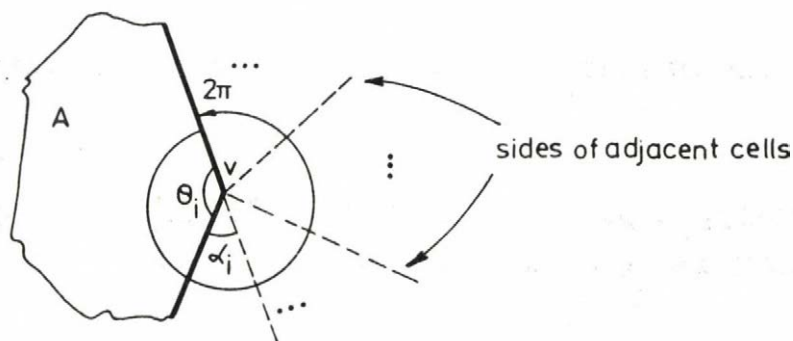


Fig. 2.4

From Fig. 2.4, it is observed that the perfect coupling among cell "A" and all other adjacent cells coincident with it in vertex "V" occurs when the expression:

$$m_i \theta_i = 2\pi \quad (2.4.6)$$

is fulfilled for positive and integer values of " m_i ".

If Exp. 2.4.6 is generalized for all other vertices of

the cell "A", then the optimal coupling in all possible directions of the cell "A" with those cells surrounding it will take place.

Therefore, Exp. 2.4.6 for each vertex is:

$$\begin{array}{lll} m_1 \Theta_1 = 2\pi & \text{for vertex 1} & (2.4.7) \\ m_2 \Theta_2 = 2\pi & \text{for vertex 2} & \\ \vdots & \vdots & \\ m_n \Theta_n = 2\pi & \text{for vertex n} & \end{array}$$

Adding these expressions we obtain:

$$m_1 \Theta_1 + m_2 \Theta_2 + \dots + m_n \Theta_n = 2\pi n \quad (2.4.8)$$

where the values m_1, m_2, \dots, m_n must be integer and positive.

Obviously, Exp. 2.4.8 must be fulfilled for optimal coupling among cells in all directions.

If the condition imposed by Exp. 2.4.6 is analyzed, we can conclude that any non-convex cell is excluded automatically for optimal coupling in all directions.

Consider a vertex "V" of an arbitrary concave cell "B" /Fig. 2.5/.

From Figure 2.5, it can be seen that it is not possible to fulfil the Exp. 2.4.6 for optimal coupling among equal-shape cells. In this case:

$$2\pi - \Theta_i \leq \Theta_i$$

since $\Theta_i > \pi$; then:

$$2\pi \leq 2\theta_i$$

$$\frac{\pi}{\theta_i} \leq 1 \quad (2.4.9)$$

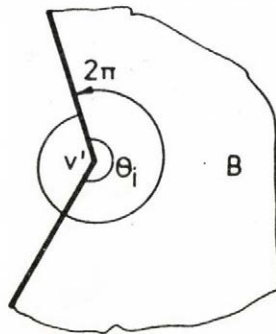


Fig.2.5

On the other hand, from Exp. 2.4.6 we have:

$$m_i = \frac{2\pi}{\theta_i}$$

$$m_i = 2\left(\frac{\pi}{\theta_i}\right) \quad (2.4.10)$$

Taking into account Exp. 2.4.9, there are only two values for " θ_i " which fulfil it in order to get integer and positive values for " m_i ", namely:

$$- \text{ if } \frac{\pi}{\theta_i} = 1, \quad \text{that is: } \theta_i = \pi$$

In this case the vertex "V" disappears, it then not being necessary to consider the coupling condition with respect to it.

$$- \text{ if } \frac{\Pi}{\theta_1} = \frac{1}{2}, \text{ that is: } 2\Pi = \theta_1$$

In this case the angle " θ_1 " also disappears and with it the cause which gives rise to this analysis. Thus, the cell could not be concave.

Exp. 2.4.5 and 2.4.8 form a simultaneous algebraic system with 2 equations and "n" unknowns, i.e.:

$$\begin{aligned} m_1\theta_1 + m_2\theta_2 + \dots + m_n\theta_n &= 2\Pi n & (2.4.11) \\ n\Pi - (\theta_1 + \theta_2 + \dots + \theta_n) &= 2\Pi \end{aligned}$$

with the condition that m_1, m_2, \dots, m_n be integer and positive.

From this, it can be established that any n-sided convex cell which fulfills Exp. 2.4.11, couples perfectly in all directions with all adjacent cells surrounding it, satisfying thus in the first instance the Exp. 2.4.2.

Exp. 2.4.7 above states the conditions for optimal coupling of cell "A" with all cells surrounding it in all possible directions.

Since by hypothesis $A_A \ll A_S$, then any cell adjacent to cell "A" surrounding it in any direction requires itself also to fulfil the Exp. 2.4.7. Hence, we can conclude that Exp. 2.4.8 is fulfilled only in case of regular polygons, i.e. with those polygons with equal inner angles " θ_i " and with equal sides " ℓ_i ". Only in these cases, will every cell of the useful surface "S" find the same geometric conditions than the cell "A" in all directions and in any place inside it.

Therefore, only cells with shapes of regular polygons are considered. In these cases:

$$\theta_1 = \theta_2 = \dots = \theta_n = \theta \quad (2.4.12)$$

If Exp. 2.4.6 is fulfilled for one vertex, it is fulfilled for any other vertex of the polygon, thus:

$$m_1 = m_2 = \dots = m_n = m \quad (2.4.13)$$

Substituting Exp. 2.4.12 and Exp. 2.4.13 in Exp. 2.4.8 we have:

$$nm\theta = 2\pi n$$

$$\theta = \frac{2\pi}{m} \quad (2.4.14)$$

where "m" is integer and positive.

Substituting Exp. 2.4.12 and Exp. 2.4.13 in Exp. 2.4.5 we have:

$$n\pi - n\theta = 2\pi \quad (2.4.15)$$

Substituting now Exp. 2.4.14 in Exp. 2.4.15:

$$n\pi - n \frac{2\pi}{m} = 2\pi$$

$$mn - 2m = 2n$$

$$m = \frac{2n}{n-2} \quad (2.4.16)$$

The cell is a closed geometric figure, therefore the minimum number of sides it can have is $n=3$. This is also evident, since "m" in Exp. 2.4.16 does not acquire integer and positive values for $n=0$, $n=1$ and $n=2$.

Therefore, the definitive new condition is:

$$m = \frac{2n}{n-2} \quad n \geq 3 ; \quad (2.4.17)$$

$m > 0$ and integer

This means that any n -side regular cell which fulfills the Exp. 2.4.17 and gives integer and positive values for " m ", couples perfectly in all directions with all cells surrounding it satisfying in the first instance the condition stated in Exp. 2.4.2.

Regular polygons which fulfil the Exp. 2.4.17 are now investigated.

If $n=3$	(equilateral triangle)	then	$m=6$	(*)
$n=4$	(square)		$m=4$	(*)
$n=5$	(regular pentagon)		$m=\frac{10}{3}$	
$n=6$	(regular hexagon)		$m=3$	(*)
$n=7$	(regular heptagon)		$m=\frac{14}{5}$	
$n=8$	(regular octagon)		$m=\frac{8}{3}$	
\vdots			\vdots	

Generalizing, the value to which " m " approaches when the number of sides of the polygon " n " approaches to ∞ is calculated.

From Exp. 2.4.17:

$$\lim_{n \rightarrow \infty} m = \lim_{n \rightarrow \infty} \frac{2n}{n-2} = \lim_{n \rightarrow \infty} \frac{2}{1-\frac{2}{n}}$$

Therefore:

$$\lim_{n \rightarrow \infty} m = 2 \quad (2.4.18)$$

Analyzing the values previously found for " m ", it is observed that those marked with (*) fulfil Exp. 2.4.17.

From these, when $n=6$, the value $m=3$ is obtained; thus, from the limit value given by Exp. 2.4.18 we can conclude that no polygon with more than 6 sides would fulfil the Exp. 2.4.17, since no other integer number exists between the integer numbers 2 and 3.

Based on the first criterion stated above and with the hypotheses pointed out through the previous analysis, it has been found that the optimal shape for the cell is one of the following geometric shapes /Fig. 2.6/.

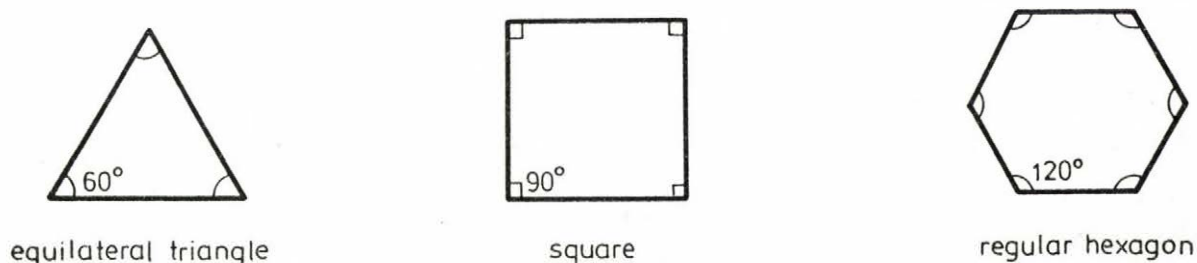


Fig. 2.6

Cells with geometric shapes shown in Fig. 2.6 are only those cells which permit an optimal coupling among them in all directions and in any place within the plane useful surface "S".

2. Now the cells of the border will be analyzed

Since the definitive optimal shape of the cell depends on the optimal shape of the useful surface "S", and at the same time the shape of the useful surface "S" depends on the shape of the cells which subdivide it optimally, then a tradeoff is required. Thus, the surfaces created when the cell shapes previously selected are combined into a complete structure are analyzed.

To carry out this analysis in most general form, we consider now how the cells conform to the edges of any

plane surface in any two possible dimensions. First of all, the corners of those surfaces composed with cell shapes selected above are analyzed.

Obviously, only the most regular surfaces which offer the most simple solutions are considered, and in no way those surfaces considered as capricious which are excluded by themselves. This analysis also considers only those surfaces having the most logical or simplest arrangement of the cells.

Since in this stage the analysis is carried out considering the coupling among cells in all directions, and in that case equilateral triangular cells form always regular hexagonal cells /Fig. 2.7/, then our analysis is carried out only referred to square and regular hexagonal cells. The analysis for regular hexagonal cells is also applicable to equilateral triangular cells.

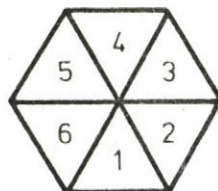


Fig. 2.7

Fig. 2.8 illustrates two corners belonging to two useful surfaces composed by cells with regular hexagonal and square shapes.

With regular hexagonal cells /Fig. 2.8a/ there are basically three different possibilities to form the corner of the surface according to the side chosen, that is, side A, side B or side C. With square cells /Fig. 2.8b/ there is only one possibility to form the corner.

corner belongs to both vertical and non-vertical sides, giving place to ambiguities in defining the coordinates of points belonging to it using a Cartesian rectangular coordinate system. Thus, this array requires a complex coordinate system to define each point inside it without ambiguities.

Array with square cells

- Both sides of this array are continuous, unique and equal /rectilinear/.
- There is ambiguity only in the corner itself. However, its rectangular coordinates can be defined uniquely because in a two dimensional coordinate system a single point is completely defined with only two coordinates.

2. Identification of a particular cell in the array

Regular hexagonal cells

- To identify each cell in the array a complex coordinate system is required.

Square cells

- The unambiguous identification of cells in the array, may be easily done by means of a Cartesian rectangular coordinate system.

3. Characteristics of the cell itself as a part of the whole surface

Regular hexagonal cells

- A regular hexagonal cell with side length " l " has larger area than a square cell with the same side length, thus requiring a higher number of information bits to define all the points included within it. Possibly for this reason together with the optimal

they are coupled in all directions. Thus, the hexagonal cell /and the regular triangular cell when they are oriented in all directions forming regular hexagonal cells/ is excluded as the optimal alternative.

Now then, as points in any plane surface are unequivocally defined with only two coordinates, the coupling of equal cells in only two directions is considered as a special case. For convenience and because of the advantages offered from the mathematical point of view, the analysis is carried out in two directions parallel with the axis of a Cartesian rectangular coordinate system.

With this new freedom degree more different cell shapes may be coupled optimally e.g. all cells having two parallel sides and even symmetry with respect to an axis parallel to them passing through the geometric center of the cell /cases a,b and c in Fig. 2.9/; or all cells with arbitrary shape but which can be coupled optimally to form a rectangle /cases e and f in Fig. 2.9/; etc.

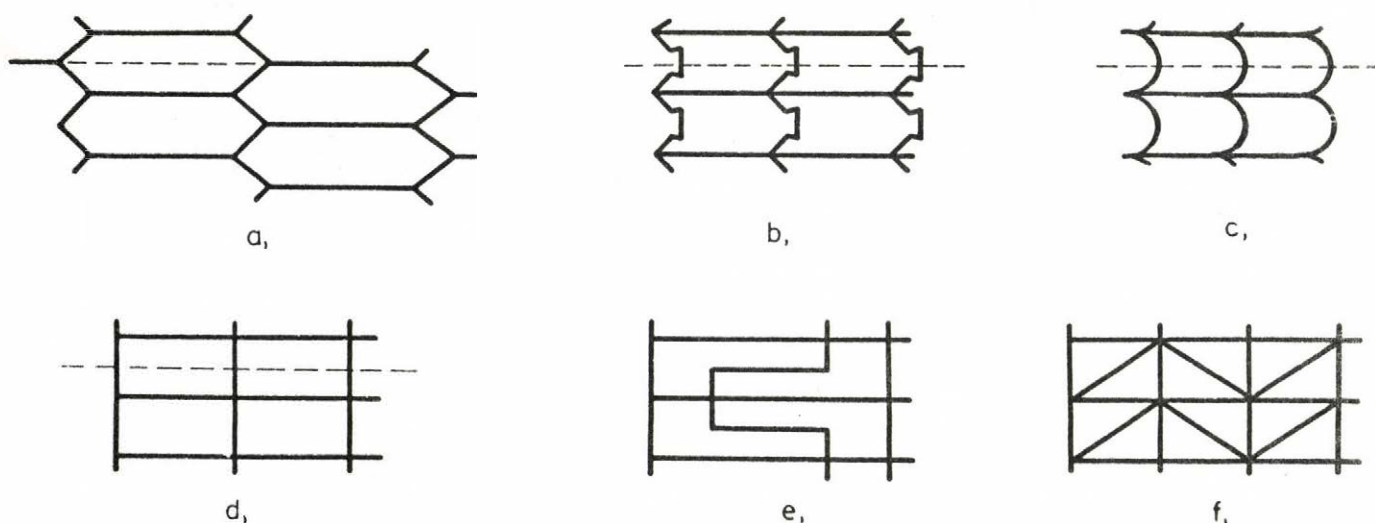


Fig. 2.9

Arrays of cells a, b and c in Fig. 2.9 are excluded as possible alternatives for similar reasons than those pointed out previously for regular hexagonal cells. However, the arrays e and f fulfil the condition of optimal coupling for getting the optimal shape for the useful surface.

2.4.3.2 Justification of Criterion 2

The second criterion pointed out above has been established to overcome the special situation of arrays such as the e and f ones in Fig. 2.9. It is motivated by practical objectives, thus the justification of this criterion is based on the actual consequences received when it is not taken into account.

The absence of the second criterion would imply:

1. To know previously the cell orientation to settle in place with reliability the information required for getting totally comprehensible pictures.
2. An undesirable dependence between the aspect of the information to be settled inside each cell and the place where it is located within the useful surface.
3. The coding and handling of a certain amount of additional information bits /at least one/, to define without ambiguity the orientation of the cell.
4. The use of special means and electronic hardware to make possible reliable transfers of information among cells /when required/, without a probable loss of information and the security that the picture will remain completely comprehensible.

Taking into account Criterion number 2, many cell shapes which fulfil the necessary condition for optimal coupling in two directions are automatically neglected themselves,

because they require a change in orientation in order to get it /Fig. 2.10/. However, that is not the situation of the cells with square or rectangular shape /nor that of those with hexagonal shape already neglected in previous analysis/ which do not require change in orientation for optimal coupling.

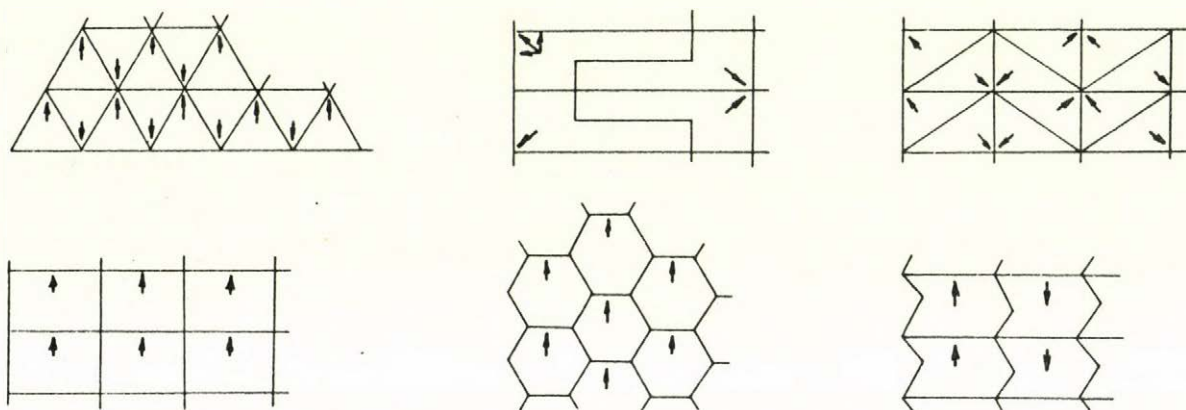


Fig. 2.10

Thus, after this analysis, we can conclude the following:

The square shape is the optimal geometric shape for the cell with the special case of rectangular shape when the optimal coupling is required in only two perpendicular directions. These cell shapes, in turn, give also the optimal shape for the useful surface "S", thus in the most general case, the square shape can be considered in similar form as the optimal geometric shape for the useful surface, with the special case of rectangular shape when other reasons such as aesthetic appearance, ergonomic requirements, special uses, etc. are obeyed.

These shapes for both the cell and the useful surface permit:

1. The use of the whole useful surface "S" to display visual information.

2. To show this visual information without redundancies and ambiguities.
3. An easy handling of the cells and easy processing of the information to be settled inside it with a simpler hardware.
4. A logical ordering of cells within the surface for easy identification.
5. To use a minimal number of information bits to define the visual information within the cell.
6. To get good aesthetic appearance.
7. To get good ergonomic characteristics.

The previous analysis supports mathematically the intuitive wide use of the square and rectangular cell in representing the visual information required by many applications in the most diverse fields of science and technology.

2.5 Common characteristics of drawings in process control applications. Graphical requirements

From the topological point of view, and analyzing the most general cases encountered in practice, there can be enumerated some common characteristics for most drawings required to control and monitor industrial processes, namely:

1. They are bidimensional drawings.
2. They can be drawn by using rectilinear segments only.
The use of segments of arcs and regular or irregular curve shapes are not required.
3. Most of drawing can be drawn using merely horizontal and vertical straight lines, not requiring at all those with arbitrary slopes.
4. They do not need to use several types of straight lines, two types being sufficient for most applications.

5. They are non-scaled drawings, thus they do not require quotas or fixed dimensional specifications for each component which composes them. Thus, they can be uniformly distributed maintaining some aesthetic form.
6. When they are drawn, there is no relative change in position among the elements or components which compose them, that is, the relative position among components is invariable after it is designed.
7. The location of components in drawings is not rigorous.
8. They can be drawn using only:
 - alphanumerical characters
 - symbols
 - limited graphics
9. They have no intersections of many lines in any one point. Moreover, they require only a reduced number of different types of intersections.
10. There are no superpositions of different types of graphical elements to compose the drawings.
11. They have no high density of graphical information.
12. They can be drawn by parts, that is, by areas, sections, zones, etc. Each one can be independently considered having the possibility to be adequately joined to the succeeding ones as they are actually distributed in the industrial plants.
13. In the same area, zone, etc. of the plant are encountered commonly several different types of networks /electrical, pneumatical, etc./ Each one of them can be graphically isolated from the other types, it being possible to consider them as independent single drawings though they are actually located in the same physical place.
14. Each drawing, independently of the type of network it represents, can be considered in general as composed by elements each one having two different natures: one

considering the element or component itself, being necessary only to show their existence /a valve, a measuring instrument, etc./, and the other considering the state of the element or component along time, that is, if the valve is opened or closed, the actual value measured by the measuring instrument, etc.

15. They do not require shading or grey tones. Moreover, they can be shown always, unambiguously, on a dark background.
16. They do not necessarily require windowing, clipping, scaling, transformations, and other sophisticated facilities commonly needed by other application fields.
17. They do not need to be displaced along the medium once they have been created; they remain always in the same relative position with respect to adjacent drawings.
18. They are always shown as a whole and never partially once they have been created.
19. They are composed by a limited number of different types of components which generally are present in quantities higher than 3 or 4, if no more. Seldom a particular type of component is encountered in quantities of 1 or 2 units in a whole industrial process.

Taking into account the common characteristics enumerated above and analyzing them, we can get the requirements for the best solution of an interactive graphic display for process control application.

So far, we have found as the best solution when we compose the drawings, the subdivision of the useful surface "S" in rectangular cells arranged uniformly /See Sect. 2.3 and Sect. 2.4/. Then, obviously all our analysis is supported considering this basic criterion.

Since the common characteristics given above are of different

nature, some criteria are established starting from their independent analysis. First of all, the graphic entities most commonly used to compose the drawings and enumerated in characteristic 8 above are analyzed. However, other characteristics from those formerly exposed will also be considered simultaneously when required, in order to arrive at best results.

2.5.1 Alphanumeric characters

As alphanumeric characters are meant the letters of a given alphabet, the numerals and the punctuation marks commonly used in editing.

Note: In all our analysis we will consider the use of the English alphabet, because of its universal diffusion around the world and its wide use in this technique.

2.5.1.1 Letters

Letters are used in this application mainly:

- in the name of each drawing to facilitate to operators its identification, that is, as a title,
- in tables, list, etc. identifying items, production unities, plants, etc.,
- in legends or inscriptions to identify symbols, specific conditions of a given drawing, etc.,
- to specify physical measuring units,
- as a literal code to join succeeding drawings,
- as a part of complex symbols,
- as literal codes for the identification of equal symbols,
- and other uses of less importance.

Letters can be capital and lower-case. In this application,

analyzing the possible uses of letters detailed above, lower-case letters could be required possibly only to specify the physical measuring units. However, this is not worthwhile when they can be specified also, with good results, by using capital letters. This does not involve misunderstandings or loss of information and represents a considerable simplification in the whole system.

Then, we can point out that the 26 capital letters of the English alphabet fulfil in the first instance all the requirements for texts in process control applications. Numerals and punctuation marks are dealt with later.

The selected font must include letters which guarantee optimal legibility. Moreover, they are required always with the same orientation and with only one size.

Discussion about the selection of the character font /which includes letters, numerals and punctuation marks/ is not included here, because in some measure it falls out of the scope of the present work. Many studies have been realized about this subject which in one form or other are published in many technical articles of the open literature[137,257,258].

2.5.1.2 Numerals

Numerals are used in this application mainly:

- in consecutive numbering of the drawings,
- in tables, list, etc. indicating some data, values, etc.,
- to indicate the measured value of measuring instruments, settings, etc.,
- as a part of complex symbols,
- as codes for the identification of equal symbols,
- to indicate the time, if required,
- and other uses of less importance.

Referring to numerals, we can point out the following:

- Generally the values handled in this application do not need to have more than 6 significant figures which it is the accuracy obtained in most of the measuring instruments used in industry. However, this is not definitive.
- When they are used together with letters, they always occupy fixed places in the context, that is, preceding a physical measuring unit, following a letter in a symbol code, a name in a table or list, etc. An appropriate convention can also be established using different colors or a discrete separation between them.
- When they are used without letters, i.e. alone representing numerical values, they are written in particular places on the screen as part of a table, a drawing, etc., where they are easily identified as numbers.
- They can have always the same orientation and size.
- They could be decimal fractional /non-integer/ numbers. In this case a reduced number of decimal places will be sufficient to express them.
- The font selected must ensure optimal legibility.

From these observations, it is evident that it is never necessary to define the character "ø" to differentiate the letter "O" from the numeral zero. They can be represented by the same character without possibility of ambiguity or misunderstanding.

2.5.1.3 Punctuation marks and editing symbols

The requirements for editing in this application are reduced, therefore at first glance the totality of punctuation marks commonly used in alphanumerical displays [24,149,150] is not necessary. Nevertheless, the use mainly of the 6-bit ASCII character code is recommended due to its frequent

application, its wide diffusion and general acceptance.

Additionally, some other characters and editing symbols not included into the ASCII code are proposed in order to fill some specific needs of this application.

The selection of those most useful punctuation marks is intimately related with the analysis of the physical measuring units and their requirements.

We have found some general characteristics to all physical measuring units which could carry us later to notable simplifications in creating the pictures. Analyzing a relatively high number of them, belonging to different fields of science [310], we can conclude the following:

- They can be expressed always by letters of the English alphabet.
- They can be in most cases abbreviated using some letters and a reduced number of special punctuation marks or symbols.
- Their abbreviations do not need to be ended with period.
- Some units are represented universally by a letter of the Greek alphabet, but they can also be always expressed with letters of the English alphabet (Ω : OHM).
- In all cases they can be written in singular number.
- They can be required always with a sole orientation and only one size.

Six-bit punctuation marks and symbols most useful in this application for editing, are tabulated in Table 2.5.1 indicating some particular uses.

Table 2.5.1

Most useful symbols and punctuation marks for editing in this application

Item Number	Symbol	Designation	Uses
1	%	per cent	- to express values in per cent
2	:	colon	- to compose particular vertical dotted lines - as separator when the time is required 12:22:15
3	-	hyphen	- for compound units (KG-M) - to create particular horizontal dashed straight lines - for the minus sign in signed numbers or mathematical expressions
4	.	period	- as decimal or lexical point
5,6	[and]	square brackets	- in chemical radicals - in texts
7	&	and /ampersand/	- in texts
8	,	comma	- in texts
9,10	(and)	parenthesis	- in chemical radicals - in texts
11	+	plus sign	- in mathematical expressions - to indicate positive signed numbers
12	@	at symbol	- in computer programming languages
13	*	asterisk	- in computer languages - as indicator

Table 2.5.1 /Continuation/





Item Number	Symbol	Designation	Uses
14	"	quotation mark	- in texts
15	/	slanted bar	- as parenthesis in some cases - as divide sign to represent either some units (KM/H) or fractional numbers
16	=	equality sign	- as direct current's symbol - in mathematical expressions
17	!	exclamation mark	- as factorial symbol - in texts
18	?	question mark	- in texts
19		space	- in texts
20		dollar /peso/ symbol	- as monetary unit
	\$		- in computer programming languages
21	#	hash mark	- in computer programming languages
22	;	semicolon	- in computer programming languages - in texts
23	>	more than	- in Boolean and algebraical expressions
24	<	less than	- in Boolean and algebraical expressions

Note: In general, although it has not been particularly emphasized, some other punctuation marks mentioned are used also in programming languages. The period, comma and semicolon symbols are not proposed to be located in the center of the cell /coincident with the vertical symmetrical axis of the cell/ because of aesthetic reasons.

Other symbols not included into the ASCII code, but proposed specifically for this application are tabulated in Table 2.5.2.

Table 2.5.2

Other editing symbols useful for the application

Item Number	Symbol	Designation	Uses
1		degree symbol	<ul style="list-style-type: none">- for temperature units ($^{\circ}\text{C}$, $^{\circ}\text{F}$, $^{\circ}\text{K}$)The use of this symbol substitutes the letters "DEG"- to substitute the symbol "# " /hash mark/ which sometimes means "number". In this case it must be used as N°
2.		second power	<ul style="list-style-type: none">- as exponent to represent square units (CM^2). In this case this symbol substitutes the letters "SQ"- in chemical formulae (H_2O)
3		third power	<ul style="list-style-type: none">- as exponent to represent cubic units (M^3). The use of this symbol substitutes the letters "CUB"- in chemical formulae (NH_3)
4		similar sign	<ul style="list-style-type: none">- as alternating current's symbol- as the symbol of fuses

Editing symbols of the 6-bit ASCII character code with limited use in this application are: \, _ , ^ , ' , although

they can have some use in computer programming languages or in creating other symbols.

Those punctuation marks and symbols of other codes which in general are not required for editing in industrial process control applications are shown in Fig. 2.11.

$-, |, \wedge, \infty, \circ, \rightarrow, \neq, \geq, \leq, \div, \sqrt{}, ', \parallel, \perp, \equiv, !, \{, \},$
 $\&, \backslash, -, \vee, \kappa, \pm, \%, \top, \Omega, \sim, \dagger, \neg, \S, \text{ etc.}$

Fig. 2.11

However, some of them could be eventually considered in other particular uses or as symbols required by programming languages created for very specialized applications.

2.5.2 Symbols oriented to the application

The correct selection of the Symbol-set is of extreme importance. The main part of drawings will be composed by symbols, hence the importance of a good selection to fulfil all the requirements imposed by this application.

General criteria to take into account for the selection of the Symbol-set are proposed as the following:

1. They must be autodescriptive symbols, that is, their shape must indicate without ambiguities the physical component which they symbolize.
2. If possible they must be of the same shape to those universally adopted by practice or by standardization.
3. If possible they must occupy as maximum the space of a single cell to facilitate its identification and manipulation.

4. If possible, each symbol must have itself more than one use, that is, as a single symbol with two or more meanings or as a part without superposition of a more complex symbol. This would permit higher flexibility to fulfil the requirements of drawings avoiding also the use of new symbols.
5. If possible, compound symbols which occupy more than one cell must be composed by single symbols, that is, consider the usage of single symbols to compose compound symbols.
6. If possible, not require the superposition of two or more single symbols to create another single symbol. This would make the display's design more costly.
7. Horizontal and vertical symbols must be of the same shape, they must occupy equal physical space and have the same aspect ratio.
8. They must couple optimally with adjacent symbols in the direction and sense required by the drawings.
9. Their legibility must be as good as possible and they must fulfil ergonomics requirements.
10. The total number of different symbols must be minimum.

As intrinsic geometric characteristics of symbols used in this application, can be pointed out the following ones:

1. Those symbols which indicate sense, require four equal shaped symbols as maximum with different orientation each.
2. Symbols which do not indicate sense, if they are symmetrical with respect to either the horizontal or vertical axis passing through its geometric center, require as maximum two /sometimes only one/ equal symbol with two different orientations.
3. Those symbols symmetrical with respect to two perpendi-

cular axes which cross each other in its geometric center, are unique for all cases.

4. Entirely asymmetrical symbols may be required in different quantities /from 1 to 4 units/ depending on their particular shapes and uses.

2.5.3 Graphics

2.5.3.1 Straight Lines

In this application limited or restricted graphics can fulfil the general requirements of drawings.

Before all, because most drawings can be drawn using merely horizontal and vertical straight lines, we will consider only the following two types of lines for graphics: lines parallel to abscissa axis and lines parallel to ordinate axis, both in a Cartesian rectangular coordinate system. With these two straight line types, the graphic requirements can be completely fulfilled, since segments of arcs and regular or irregular curve shapes are not required. These conclusions are based on characteristics of drawings number 2 and 3 previously enumerated /Sect. 2.5/.

In a raster scan display, these two types of lines can be easily traced in two different forms:

1. Controlling opportunely the blanking and unblanking of the CRT'S electron beam /scanning approach/.
2. Using specially defined single symbols to carry out this operation.

As we already know, drawings required in this application are non-scaled, as soon as the location of symbols inside them is not rigorous /characteristics 5 and 7/. Then, we can affirm

that the use of straight lines with all possible lengths between a specified minimal and maximal length, are not necessarily required.

Taking into account also that all given long straight lines can always be traced using straight segments /or shorter straight lines/ with equal slope each and one succeeding each other, the second alternative will be preferred.

The second alternative offers the following advantages with respect to the first one:

1. It makes best use of the specific common characteristics of drawings required in this application /e.g. there is no need for a horizontal straight line in every existing scan/.
2. It requires simpler hardware.
3. It permits an appreciable data compression, saving information bits per instruction-word and consequently saving memory capacity.
4. It facilitates programming.

Thus, we must define now single symbols to trace horizontal and vertical straight lines. We have established already as one criterion for single symbols of the set, that each one must occupy as maximum the space of one cell. Then, as most of single symbols are commonly symmetrical with respect to an axis parallel to the abscissa or ordinate axis /or both/ passing through the geometric center of the symbol, we establish as the best solution, the tracing of either horizontal and vertical straight lines through the center of the cell sides. Thus, only two single symbols are required for the limited graphics, namely: single symbol to be used to draw continuous horizontal straight lines /Fig. 2.12a/ and single symbol to be used to draw continuous vertical straight lines /Fig. 2.12b/.

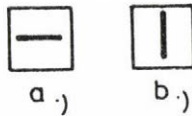


Fig.2.12

Note: Although the cells will be shown square in shape, this does not mean that we are affirming that the optimal relation between non-parallel sides in this application must be 1:1. The analysis of this problem is done later /See Sect. 2.7/. Then, the symbols in this part of the work are shown only qualitatively, there being no loss of generality in all our analysis with this consideration.

2.5.3.2 Intersections

We consider now the possible types of intersections which could take place in drawings used in process control applications /characteristic 9/. Intersections take place only in graphics and are of two general types: intersections without contact among lines and intersections with contact among lines.

1. Without contact among lines

Although the drawings are bidimensional and 3-D drawings are not required in this application /characteristic 1/, there could be the possibility of two straight lines crossing each other without contact between them. The case of more than two lines crossing each other through the space occupied by a cell is not considered, since the drawings are non-scaled /characteristics 5 and 9/. In case it occurs we can draw it always conveniently to overcome this remote possibility. Thus, the graphical representation of the possible intersection of

two straight lines without contact in a cell array, can be done in the following two manners shown in Fig. 2.13.



Fig. 2.13

With these solutions the following advantages are obtained:

- It is not necessary to define new symbols
- Either the horizontal or the vertical straight line, one of them in each case, is shortened because of its partition in two lines
- There is no ambiguity with the interpretation, that is, it represents clearly the crossing without contact between both lines.

2. With contact among lines

In this case we can find the following two possibilities:

- incomplete intersection
- complete intersection

Incomplete intersection

Incomplete intersection takes place when the tracing of one straight line does not continue beyond the contact point with the other one.

Depending on the orientation, there are four different possibilities:

- vertical straight line going downwards intersecting incompletely a horizontal straight line /Fig. 2.14a/,

- same going upwards /Fig. 2.14b/,
- horizontal straight line going towards the right-hand side intersecting incompletely a vertical straight line /Fig. 2.14c/,
- same going towards the left-hand side /Fig. 2.14d/.

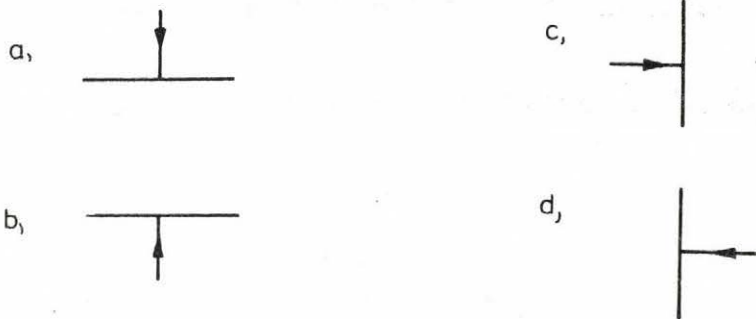


Fig.2.14

Since the symbols used to produce any straight line travel through the center of the cell either horizontally or vertically, the solution for these can be given by the following special four symbols /Fig. 2.15/:



Fig.2.15

This solution gives the following advantages:

- It is not necessary to define the tracing of straight line segments with lengths inferior to a cell side's length
- The horizontal or vertical straight line being intersected is shortened
- It will facilitate the required algorithms to analyze the intersection
- It permits drawings with good aesthetic qualities.

Disadvantage:

- four new symbols are required for restricted graphics.

Complete intersection

Complete intersection takes place when both perpendicular straight lines continue beyond the contact point.

In this case there is only one possibility /Fig. 2.16/.

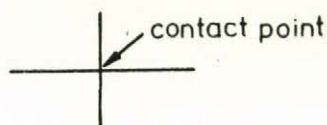


Fig. 2.16

This case could be solved using two different alternatives:

1. Creating a new symbol for the intersection. In this case, the symbol obviously is that shown in Fig. 2.17.



Fig. 2.17

This alternative gives the following advantages:

- it is necessary to use only one symbol code for the intersection
- both the horizontal and vertical lines are shortened
- the representation of the intersection occupies less physical space on the drawing with respect to other solutions, that is, only one cell.

Drawbacks of this alternative are:

- the definition of a new symbol which could be saved /it would contradict criterion 10 for the selection of the Symbol-set, Sect. 2.5.2/
- the aesthetic lack when defining colors for each horizontal and vertical segment. However, if there is a possibility of defining this symbol by means of the superposition of two symbols with only one instruction-word, this drawback could be overcome, but this would contradict criterion 6 /Sect. 2.5.2/
- the required algorithm to analyze the intersection will be more complex because of the relatively high number of different possibilities /three/
- tracing of drawings is less flexible. For example, in case of two parallel lines coming out to the right from the contact point, every possible and topologically equivalent solution must use the shaded cell shown in Fig. 2.18. If it is not so, the use of this symbol in this case is not justified.

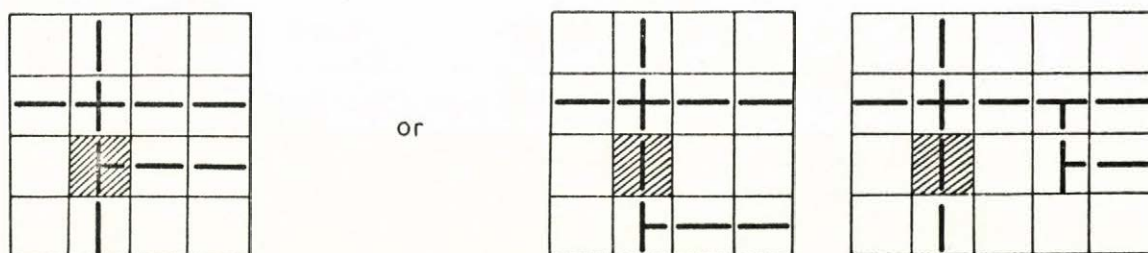


Fig. 2.18

2. Creating the complete intersection with symbols previously defined.

The symbols inside the two cells delimited with dotted lines in Fig. 2.19 are equivalent to the previous symbol defined for the complete intersection.

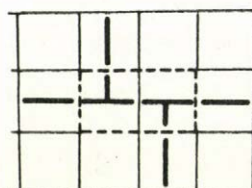


Fig. 2.19

In this case we have the following advantages:

- it is not necessary to define new symbols
- it is possible to define different colors for whichever horizontal and vertical straight lines, with acceptable result from the aesthetical point of view
- the required algorithm to analyze the intersection will be simpler /to decide from two possibilities/
- tracing of drawings is more flexible. In case of two parallel lines coming out from the contact point, proposed possible solutions do not obligate us to use a specific cell /Fig. 2.20/

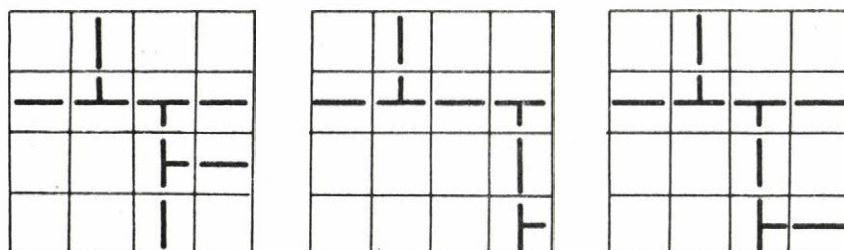


Fig. 2.20

- there is no necessity to consider any new symbol when more complex intersections take place in only one point, that is, when there are two or more nearly located parallel lines traveling in two or more directions. In all cases,

the intersection can be drawn by using symbols shown in Fig. 2.15 in straightforward form.

Drawbacks of this alternative are:

- the intersection occupies one cell more than above; if one instruction-word is required per symbol or cell, then it requires also one instruction-word more than in the previous case.

Analyzing the advantages of proposed symbols for incomplete intersection when they are used also to represent the complete intersection /which are the disadvantages when a new symbol is created for this objective/, and following the criterion for a minimum total number of symbols, we can conclude that only the four previously proposed symbols for incomplete intersections shown in Fig. 2.15, must be included into the Symbol-set for this application.

2.5.3.3 Corners

Corners are produced when a horizontal straight line finishes in the same point in which a vertical straight line begins, or viceversa. In this application, by the characteristic 3 of drawings, corners are produced by straight lines forming a right angle between them.

To avoid the definition of straight lines with lengths lesser than the length of a cell side /which have not a true utilization in non-scaled drawings/ and the superposition of symbols, the corners must be solved by means of specific symbols representing them. Those proposed are shown in Fig. 2.21.

In general, the characteristics of both segments constituting the corners will always be the same, that is, color, type of line, etc., which facilitate programming.



Fig. 2.21

2.5.3.4 "Cutting" of symbols

Based on the general criteria proposed previously to take into account in selecting the Symbol-set and the geometric characteristics pointed out formerly /Sect. 2.5.2/ a promising solution is proposed in designing the Symbol-set to be used in this application. Results are compatible with the selection of symbols carried out previously for the restricted graphics required.

The solution proposed consists in creating particular single symbols from which simpler symbols could be obtained by means of "cutting" them by hardware in the display.

As a result of this, from each one of these particular single symbols, a great number of new single symbols can be obtained, depending on the different types of "cutting" proposed.

Two types of "cuts" are proposed:

- cuts type 1, designated "cuts in quadrants" and
- cuts type 2, designated "cuts in level".

In Fig. 2.22a, the shaded part of the original single symbol shown is considered as the part which results after "cutting", that is, the shaded part consists of the area occupied by the new single symbols obtained. For these kinds of regular symbols, eight different "cuts" are proposed which are

illustrated in Fig. 2.22b,c,d in case of a full cell, a cross and a rhombus /or diamond/ symbols, respectively.





































original		cuts							
		0	1	2	3	4	5	6	7
a									
b	 Cross								
c	 Rhombus								
d	 Full cell								

Fig. 2.22

In Fig. 2.23, "cuts" type 2 are illustrated. This type of cut is particularly useful in creating vertical bar diagrams rendering a precision of $\pm 0.5d$, where "d" is the width of a TV line. Moreover, it renders a diversity of new symbols from each one particularly selected for the application.

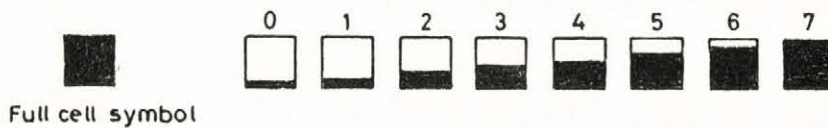


Fig. 2.23

These 16 different types of "cuts" shown above, require 4 information bits in order to identify them. They shall be designated here as "modifier" bits /See Sect. 4.4.4./

Advantages of "cuts" are:

- It permits us to render a higher number of single symbols from a given Symbol-set, i.e., from each particular single symbol stored in a ROM /or PROM/, it is possible to dispose of 16 new single symbols easily identifiable. This is equivalent to multiplying the Symbol-set by 16.
- It gives a better use to bits of the ROM, since many symbols commonly occupy a reduced number of dots when they are represented /e.g. corners only occupy approximately 1/8 of the total number of dots of the cell/.
- "Cuts" can be easily implemented by hardware in the display itself, not requiring an appreciable additional processing from the computer.

A disadvantage of "cuts" is:

- They require 4 additional bits in the display data-word and of course also, in the instruction-word which define them /See Sect. 4.4.4/.

2.5.3.5 Types of straight lines

Considering the use of color to differentiate straight lines with similar meaning and parameters /slope and length/, and taking into account characteristic of drawings number 4, the use of only two types of straight lines is proposed, namely:

- continuous straight lines
- intermittent straight lines

Intermittent straight lines can be generated in one of two different forms:

- producing the interruption of continuous straight lines by indirect means
- using special symbols

From these two alternatives, the second one could be more

advantageous, because it does not require the inclusion of new hardware. In this case the inclusion of two new symbols is required /Fig. 2.24/. Nevertheless, in case the saving of symbol codes be necessary for more important symbols, then the first alternative must be selected. In this case, one bit per instruction word will be additionally required.



Fig.2.24

If the display has not the possibility of color coding /that is, if it uses a black and white CRT/, in that case more than two types of straight lines having different width and intermittencies /dotted, dashed, etc./ could be required. Then, new symbols will be needed for each one of them, or maybe the first alternative could then be most commendable in that case.

Symbols previously proposed for intermittent lines ensure a regular spacing between every two successive short straight segments.

Taking into account characteristics of drawings numbers 5 and 7, we will analyze now the requirements for straight line lengths in this application.

2.6 Length of Straight Lines

2.6.1 Saving of information bits in straight line definition

Straight lines required for the drawings do not need to have all possible lengths in this application.

So far, we know that:

1. The intersections with contact between lines are drawn using specially defined symbols inserted conveniently everywhere the intersection takes place, breaking the corresponding straight line which is intersected.
2. The intersections without contact among lines are represented breaking also either the horizontal or vertical line.
3. The symbols for components or elements included in the drawings are logically located between two straight lines and not over them.

We have established already that those straight lines required by the application are built by joining successively a given number of special single symbols previously defined. These single symbols are of two types, one for the horizontal lines and another for the vertical ones. With this in mind, it would seem obvious that the required binary code for the complete definition of straight lines will be longer for longer lines than for shorter ones.

Thus, we shall now discuss the maximal straight line length to be defined in order to get the best results.

The criterion on which we will base our analysis will be that which permits us the maximal saving of information bits to define the straight lines with a binary code, taking into account the frequency of usage /or of occurrence/ of straight lines with different lengths.

We suppose that any horizontal or vertical straight line with length "L" cells or less within a useful surface "S", can be uniquely identified with a minimum of " $\log_2 L$ " information bits. Then, we can consider the value "L" equal to the maximal number of successive corresponding single symbols /or cells/ required to build-up the longest straight line, either horizontal or vertical. Thus, implicitly, we suppose also

that the number specified by a given code with " $\log_2 L$ " information bits, indicates the number of corresponding necessary single symbols arranged successively to build-up a given horizontal or vertical straight line.

The objective will be to demonstrate which is better for getting the greatest saving of bits per instruction-word, that is, to use always " $\log_2 L$ " information bits per instruction-word to define all straight lines /short and long straight lines/ or to use only " $\log_2 \frac{L}{t}$ " /t: integer number/ information bits for defining only those most frequently required.

Let "x" be the total number of straight lines to be traced to compose the drawings commonly required in this application. The shortest possible either horizontal or vertical straight line, would have a minimal length equal to the respective side length of the rectangular cell occupied by the respective single symbol conceived for this purpose. On the other hand, the longest possible horizontal or vertical straight line will have a maximal length equal to the width or height of the useful surface "S", respectively.

We consider that for each straight line a given instruction-word with "M" bits is required. This instruction-word will contain all that is necessary to define completely the straight line to be traced. The question is to find the minimal number of information bits "n" required to indicate the number of repetitions of corresponding single symbols needed for tracing all-length straight lines, with maximal saving of bits in the total number of repetition bits per instruction-word.

With the aim of doing our analysis independently of the lengths of both sides of the rectangular useful surface "S", we should analyze only lengths " ℓ " equal to:

$$\ell = \frac{L}{t} \quad (2.6.1)$$

where: L is the maximal-length straight line, either horizontal or vertical, which could be traced in the useful surface "S"

ℓ is the length of the straight line, either horizontal or vertical, to be traced

t is an integer number denoting the number of equal-length partitions considered in tracing the maximal-length straight line " L " ($2 \leq t \leq L$).

As we established above, both " ℓ " and " L " are expressed in number of rectangular cells with a given predetermined side-length. Then, for a given partition " t_0 ", the minimal number of information bits " n_0 " required to indicate the repetition for tracing a straight line with length " ℓ_0 " depends directly on:

$$\log_2 \ell_0 = \log_2 \frac{L}{t_0}$$

being " n_0 " the minimal integer which satisfies the expression:

$$2^{n_0} \geq \frac{L}{t_0}$$

To facilitate the analysis and calculations and for handling practical situations, we shall consider only the partitions " t " equal to:

$$t = 2^{S_n} \text{ cells ; } 1 \leq S_n \leq \log_2 L \quad (2.6.2)$$

S_n : integer number

when the quantity "L" is selected conveniently to give an integer logarithm to the base 2.

Therefore, we can point out from Exp. 2.6.1 and Exp. 2.6.2 now that:

$$\ell = \frac{L}{2^{S_n}} \quad (2.6.3a)$$

and

$$n = \log_2 \frac{L}{2^{S_n}} \quad (2.6.3b)$$

where: n is the minimal number of repetition bits required to trace a straight line with a length " ℓ " not larger than $\frac{L}{2^{S_n}}$ cells

L is the length in number of cells of the longest straight line which can be traced

S_n is an integer number in the range:

$$1 \leq S_n < \log_2 L$$

From Exp. 2.6.3b we have:

$$S_n = \log_2 L - n \quad (2.6.4)$$

We do not consider the situation when $S_n = \log_2 L$, because in this case there would be no necessity for repetition, that is, the repetition would be zero. In this case, the straight line is traced by using only the corresponding single symbols.

On the other hand, we do not consider either the situation when $S_n = 0$, because in this case $t=1$ and there is not partitioning.

Now, we suppose that from the total number of straight lines "x" necessary to build-up a given drawing, an appreciable large given number of them have a given length "ℓ" defined by:

$$\ell_s \leq \frac{L}{2^{S_n}} \text{ cells ;} \quad S_n = 1, 2, \dots, (\log_2 L) - 1$$

log L: integer number

Let "y" be the greater probability of occurrence " P_1 " of these straight lines, then:

$$P_1 = y \quad ; \quad 0.5 < y \leq 1$$

Obviously, the probability of occurrence " P_2 " of the rest of the total number of straight lines "x" with lengths "ℓ" in the range:

$$\frac{L}{2^{S_n}} < \ell \leq L$$

will be:

$$P_2 = 1 - y \quad ; \quad 0 < 1 - y \leq 0.5$$

If we suppose that those more probable straight lines require only one instruction-word each with "M" bits as maximum, from which a number "n" are destined for the repetition and "v" bits for the instruction code, the color identification and other modifiers, then the total number of bits used only in repetitions to represent the most frequent straight line lengths will be given by:

$$Z_1 = nyx \quad (2.6.5)$$

where:

$$n = M - v \quad (2.6.6)$$

On the other hand, since to represent those straight lines with lengths larger than the selected value $\ell_s = \frac{L}{2^{S_n}}$ and with probability of occurrence $P_2 = 1 - y$ are necessary more than one instruction-word per straight line, then the total number of information bits used in repetitions to represent the least frequent straight lines lengths will be given by:

$$Z_2 = pn(1-y)x \quad (2.6.7)$$

where "p" is the number of instruction-words required to represent longest straight lines using the shortest selected one with maximal length:

$$\ell_s = \frac{L}{2^{S_n}}$$

To find the value of "p", since we established previously that "n" is the number of information bits used to define the required repetition of those most frequently encountered straight lines with lengths less than a certain value, then the maximal number of cells possible to be occupied with these "n" bits is given by 2^n . On the other hand, if a straight line longer than $\frac{L}{2^{S_n}}$ has a length of $\ell = L_r$ cells, then in this case " $\log_2 L_r$ " information bits are required.

Therefore, the expression for "p" will be:

$$p = \frac{L_r}{2^n}$$

Since we consider as the most general case the set of straight lines "x" which at least include one straight line "l" with length /expressed in number of cells/ larger than the number of cells each subdivision "t" is considered to have and less than or equal to the maximal length "L", that is, in the range $2^{S_n} < \ell \leq L$ for each value of " S_n ", then the maximum value for "p" will be given by the expression:

$$p = \frac{L}{2^n} \quad (2.6.8)$$

Substituting in Exp. 2.6.8 the value "n" obtained from Exp. 2.6.3b we get:

$$p = 2^{S_n}$$

Thus, Exp. 2.6.7 will now be:

$$z_2 = 2^{S_n} n(1-y)x \quad (2.6.9)$$

Then from Exp. 2.6.5 and Exp. 2.6.9, the total number of information bits "z" required by a given drawing having "x" straight lines, when they are represented by the shorter binary code required for the number of cells included into a given segment product of a given partition of the longest straight line "L", is:

$$z = z_1 + z_2 = nyx + 2^{S_n} n(1-y)x \quad (2.6.10)$$

On the other hand, if all straight lines "x" /both short and long ones/ are represented by a unique repetition code with:

$$k_o = \log_2 L \quad (2.6.11a)$$

information bits, then:

$$z' = k_o x = x \log_2 L \quad (2.6.11b)$$

information bits are required, where:

$$k_o = M' - v \quad (2.6.12)$$

In this case there is not partitioning and all straight lines are defined always with only one instruction-word with "M" bits per instruction-word. Since we have supposed that for the corresponding instruction code, color code and other modifiers, are required a fixed specific number "v" of information bits /See Exp. 2.6.6 and Exp. 2.6.12/ and because $k_0 > n$, then we can affirm that $M' > M$.

This value "M" is limited by the length of the computer word "W" of the minicomputer destined for processing the drawings.

Thus obviously, for a given computer-word length "W", the larger the repetition code " k_0 ", the shorter the number of bits "v" which can be destined for modifiers.

Note: In this analysis we exclude the definition of straight lines with different lengths by means of different-code lengths, because this is not the most adequate and convenient practical solution from the hardware point of view for designing the control logic and buffer register's circuits.

From Exp. 2.6.10 and Exp. 2.6.11b, we can establish for a given probability of occurrence "y" that:

if $Z > Z'$ then the best solution to get maximal saving of repetition bits takes place using $\log_2 L$ -bit codes

if $Z < Z'$ then the best solution will be to use n-bit codes ($n < \log_2 L = k_0$)

When $Z = Z'$, both solutions are equivalent. To find the boundary curve between those zones wherein each code represents the best solution for a given probability of occurrence "y", the following equality must be established:



$$Z = Z'$$

Then from Exp. 2.6.10 and Exp. 2.6.11b:

$$\begin{aligned} nyx + 2^{\frac{S_n}{n}} n(1-y)x &= x \log_2 L \\ ny + 2^{\frac{S_n}{n}} n(1-y) &= \log_2 L \end{aligned} \quad (2.6.13)$$

Substituting Exp. 2.6.3b in Exp. 2.6.13:

$$y \log_2 \frac{L}{2^{\frac{S_n}{n}}} + 2^{\frac{S_n}{n}} \log_2 \frac{L}{2^{\frac{S_n}{n}}} (1-y) = \log_2 L$$

Transforming the logarithms:

$$\begin{aligned} y \log_2 L - y S_n + 2^{\frac{S_n}{n}} \log_2 L - 2^{\frac{S_n}{n}} y \log_2 L - S_n 2^{\frac{S_n}{n}} + \\ + y S_n 2^{\frac{S_n}{n}} &= \log_2 L \end{aligned}$$

Verifying:

$$y = \frac{\log_2 L - \frac{S_n 2^{\frac{S_n}{n}}}{S_n - 1}}{\log_2 L - S_n} \quad (2.6.14)$$

where:

$$1 \leq S_n \leq (\log_2 L) - 1$$

$\log_2 L$: integer number

Exp. 2.6.14 gives the corresponding limit probability of occurrence "Y", in which there is neither saving nor subutilized information bits to define all the straight lines required to build up a given drawing in a useful surface "S", having a maximal side length "L" /either horizontal

or vertical/ and for different values " S_n " defining the partitioning " t ".

From Exp. 2.6.3b, Exp. 2.6.9 and Exp. 2.6.14, the values obtained for " n ", " p " and " y " in function of the maximal length " L " for given integer values " S_n " in the range:

$$1 \leq S_n \leq (\log_2 L) - 1 ; \quad \log_2 L : \text{integer number}$$

are given in Table 2.6.1.

From Table 2.6.1 has been prepared the Table 2.6.2 for different values of the maximal length " L ".

Because they have no practical significance here, those values for " L " less than 16 cells and greater than 256 cells have not been considered in this analysis.

Characteristics " y " vs. " S_n " for different values " L " are shown in Fig. 2.25 /from values of Table 2.6.2/.

Note: Characteristics for other values for " L " different to those considered here in the range $16 < L < 256$, can be found by logarithmic interpolation.

In Fig. 2.25, the zone above each characteristic indicates those cases when it is worthwhile to use " n " repetition bits ($n = \log_2 \frac{L}{S_n}$) for a given high frequency /or probability/ of occurrence² of shorter straight lines in order to save information bits per instruction-word. The zone under the corresponding characteristic indicates the values of the higher probability of occurrence of shorter straight lines, for which it is more advantageous to use $\log_2 L = k_0$ information bits per instruction-word for every partition $t = 2^{S_n}$. The characteristics themselves represent the values for which both solutions are equivalent, i.e. they show, for each value

Table 2.6.1

S_n	n	y	p words
1	$\log_2 L - 1$	$\frac{\log_2 L - 2}{\log_2 L - 1}$	2
2	$\log_2 L - 2$	$\frac{\log_2 L - 8/3}{\log_2 L - 2}$	4
3	$\log_2 L - 3$	$\frac{\log_2 L - 24/7}{\log_2 L - 3}$	8
4	$\log_2 L - 4$	$\frac{\log_2 L - 64/15}{\log_2 L - 4}$	16
5	$\log_2 L - 5$	$\frac{\log_2 L - 160/31}{\log_2 L - 5}$	32
6	$\log_2 L - 6$	$\frac{\log_2 L - 384/63}{\log_2 L - 6}$	64
7	$\log_2 L - 7$	$\frac{\log_2 L - 896/127}{\log_2 L - 7}$	128
...			
$(\log_2 L) - 1$	1	$\frac{L - 2\log_2 L}{L - 2}$	$\frac{L}{2}$
0	$\log_2 L$	$1 - \frac{1}{\log_n 2 \log_2 L}$	1

Table 2.6.2

S_n	L=16 cells			L=32 cells			L=64 cells			L=128 cells			L=256 cells		
	n	y	p	n	y	p	n	y	p	n	y	p	n	y	p
1	3	0.666	2	4	0.750	2	5	0.800	2	6	0.833	2	7	0.857	2
2	2	0.666	4	3	0.777	4	4	0.833	4	5	0.866	4	6	0.888	4
3	1	0.571	8	2	0.785	8	3	0.857	8	4	0.893	8	5	0.914	8
4	-	-	-	1	0.733	16	2	0.866	16	3	0.911	16	4	0.933	16
5	-	-	-	-	-	-	1	0.838	32	2	0.919	32	3	0.946	32
6	-	-	-	-	-	-	-	-	-	1	0.904	64	2	0.952	64
7	-	-	-	-	-	-	-	-	-	-	-	-	1	0.944	128
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

0	4	0.638	1	5	0.711	1	6	0.759	1	7	0.793	1	8	0.819	1
---	---	-------	---	---	-------	---	---	-------	---	---	-------	---	---	-------	---

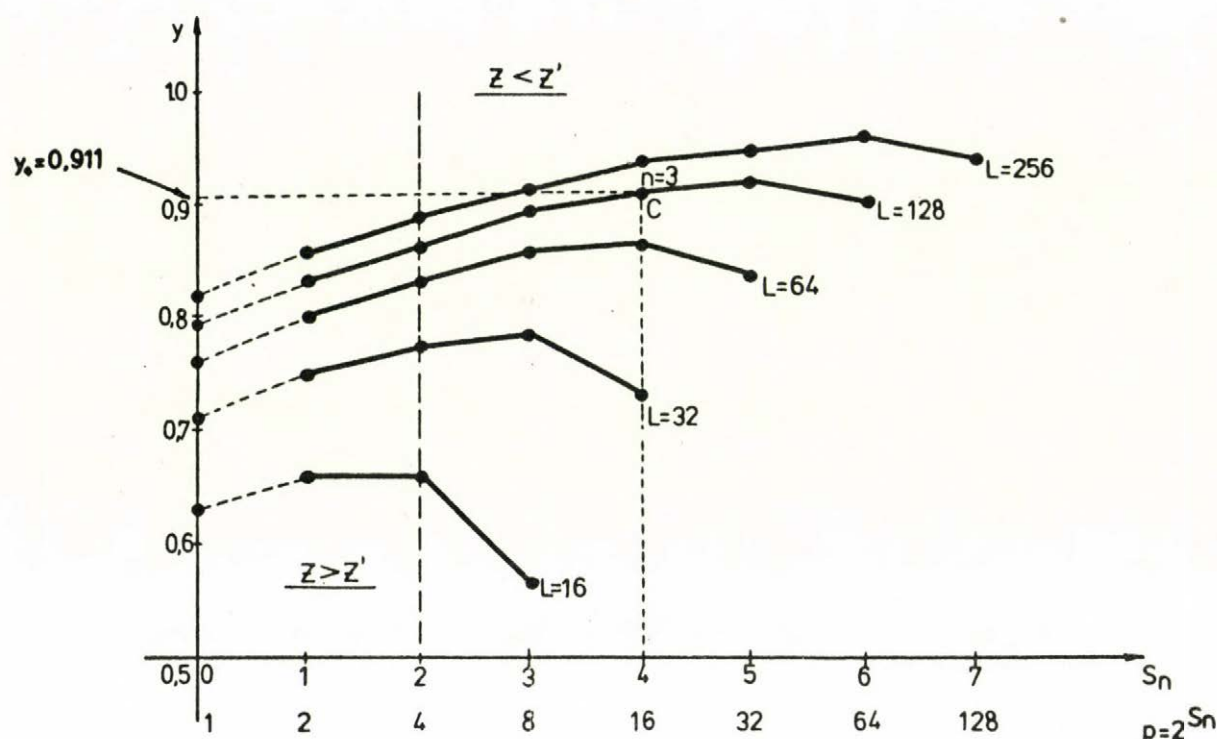


Fig. 2.25

of the maximal straight line length " L ", the limit probability of occurrence " y " at which it is worthwhile to effectuate a given partitioning $t = 2^{S_n}$ in order to save information bits using the repetition facility. The maximal value for " n " /i.e. k_0 /, using a minimal number of words in each case, is " $\log_2 L$ " bits when " $S_n=0$ " /when there is no partition/. Obviously, a considerable possible reduction of repetition bits could only take place when the probability of occurrence " y " for a given partitioning $t=2^{S_n}$ is relatively larger than its value when $S_n=0$.

Illustrating with one example, we analyze the point "C" over the characteristic for $L=128$ cells. This means that only in

that case when straight lines with length equal to or less than $\ell=2^{S_n}=2^4=16$ cells /Exp. 2.6.2/ appear with a high frequency of occurrence "y" equal to or larger than 0.911, is it worthwhile to use a shorter code with:

$$n = \log_2 \frac{L}{2^{S_n}} = \log_2 \frac{128}{16} = 3$$

information bits for repetition, in order to save bits in the instruction-word destined for this purpose. Then, the probability of occurrence of any straight line with length $16 < \ell \leq 128$ cells will be $1-y = 0.089$. In this case the maximum number of instruction-words required will be $p=2^{S_n}=2^4=16$ /Exp. 2.6.9/. To reduce the number of computer words required when the worst case takes place, it is worthwhile to sacrifice in repetition at least one bit more per instruction-word, that is to use $n=4$, thus reducing the value for " S_n ". In this case $S_n=3$ /Exp. 2.6.4/, being required as maximum only 8 words /Exp. 2.6.9/ when a straight line larger than 64 cells /Exp. 2.6.3a/ must be created.

Now then, since the number of instruction-words "p" depends exponentially on the selected value for " S_n " /Exp. 2.6.9/, obviously it is convenient to limit the maximal number of instruction-words for low values of "n". From Exp. 2.6.4, we can see that for given values of "L" and for a limited number of permitted instruction-words "p", the minimal value for "n" is also implicitly limited.

For each value "L", there is a value for " S_n " at which the required frequency of occurrence is a maximum. This case takes place always for $n=\log_2 L - S_n=2$.

In this case saving of repetition bits per instruction-word is appreciable, but the number of words required in the worst case given by Exp. 2.6.9:

$$p = 2^{(\log_2 L) - 2}$$

is practically prohibitive in representing long-length straight lines in drawings.

All this implies that a trade-off is required in order to get the best results for the generality of drawings used in industrial process control applications.

To choose the best solution, we have analyzed some typical drawings already designed to be used in the application dealt with. From these, depending on the frequency of usage of straight lines having different lengths, we have prepared the Table 2.6.3.

All data included in Table 2.6.3 have been tabulated according to their relative length respect to the larger side-length "L" of the rectangular useful surface "S" wherein the drawings were drawn. In all cases the longest side has been the horizontal one. With this artifice, the data have been considered independently of the dimensions of the cell, which have not yet been discussed so far.

Thus, straight lines have been subdivided in three groups:

1. Horizontal and vertical lines with relative length less than or equal to one-fourth of the horizontal side-length "L" of the rectangular useful surface "S", that is:

$$0 < \ell \leq 1/4 L \quad \text{/short straight lines/}$$

2. Lines with relative length " ℓ " in the range:

$$1/4L < \ell \leq 1/2L \quad \text{/medium-length straight lines/}$$

Table 2.6.3

No.	Total number of straight lines	% of straight lines with lengths in the range			Observations
		$0 < l \leq 1/4L$	$1/4L < l \leq 1/2L$	$1/2L < l \leq L$	
1	156	98%/153/	2%/3/	0%	Drawing used in pump stations [310]
2	193	100%/193/	0%	0%	Drawing used in power networks [310]
3	189	98.4%/186/	1.6%/3/	0%	Drawing used in pump stations [310]
4	56	98.2%/55/	0%	1.8%/1/	Drawing used in power networks [31]
5	122	97.5%/119/	2.5%/3/	0%	Drawing used in chemical plants [96]

3. Lines with relative length " ℓ " in the range:

$$1/2L < \ell \leq L$$

/long straight lines/

Note. Obviously, taking into account the considerations pointed out above, horizontal and vertical straight lines would have only equal actual length expressed in number of cells, when the cell is square in shape. In any other case, the actual length of horizontal and vertical straight lines created with equal number of cells arranged successively will be different.

Observations about the data included in Table 2.6.3 are:

1. Since the actual drawings can be built-up in many different forms when different distribution is given to those graphical elements which compose them, the tabulated data are only indicative and not completely definitory. However, all lines were considered with those lengths with which they have been drawn in the drawing, although in some cases a better solution could be attained.
2. Everywhere intersections took place, we considered them as a breaking point of a long line in two short lines.
3. Short straight segments used for creating given compound symbols in drawings were not considered as independent straight lines, but only as a part of those specific symbols. Thus, they were not tabulated.
4. It was not desirable to consider the analysis of short segments with other length than $\frac{L}{8}$, $\frac{L}{16}$, and so on, because of the difficulties reached with the accuracy of drawings analyzed. Moreover, the difference in number of cells between short line lengths themselves is not so significant as the difference between short lines and medium or long ones. In these cases also, because the partitions " t " are very short, then " n " has too low a value, requiring for a given maximal length " L " a large

number of instruction-words $p = \frac{L}{2^n}$, this being prohibitive in defining long straight lines.

From Table 2.6.3 we can appreciate that in the application considered, the probability of occurrence of straight lines with length in the range:

$$0 \leq l \leq 1/4L$$

is considerably higher than in the other two cases analyzed.

Then, from Exp. 2.6.1 and Exp. 2.6.2, the partition "t" and the value " S_n " will be:

$$t = 2^{S_n} = 4$$

$$S_n = 2$$

Exp. 2.6.3b, which gives the number of repetition bits will be then:

$$n = \log_2 \frac{L}{2^{S_n}} = \log_2 \frac{L}{4}$$

and the maximal number of instruction-words required in the worst case /from Exp. 2.6.9/, will be:

$$p = 2^{S_n} = 4 \text{ instruction-words.}$$

Thus, we can establish the following criterion:

Criterion 1

To construct the limited graphics required in drawings used in industrial process control applications by means of the repetition of the two single symbols proposed for this purpose, only:

$$n = \log_2 \frac{L}{4}$$

information bits per instruction-word are sufficient as maximum, where "L" is the longest straight line /either horizontal or vertical and expressed in number of cells/, equal to the longest side of the rectangular useful surface "S" wherein the drawings are drawn.

In this case, the maximal number of instruction-words required in the worst case /when $\ell=L$ / is only 4 instruction-words.

2.6.2 Saving obtained in repetition bits expressed in percent

The expression which would give the saving obtained in repetition bits "r" expressed in percent when partitioning is considered, is:

$$r = 1 - \frac{n}{k_0} \times 100 \quad (\%) \quad (2.6.15)$$

Substituting Exp. 2.6.3b and Exp. 2.6.11a in Exp. 2.6.15 we have:

$$r = 1 - \frac{\log_2 \frac{L}{S_n}}{\log_2 L} \times 100 = 1 - \frac{\log_2 L - S_n}{\log_2 L} \times 100$$

$$r = \frac{S_n}{\log_2 L} \times 100 \quad (\%) \quad (2.6.16)$$

The maximal saving should take place when " S_n " is maximum. In this case:

$$S_n = \log_2 L - 1$$

Substituting this in Exp. 2.6.16 we have:

$$r = \frac{\log_2 L - 1}{\log_2 L} \times 100 \quad (\%)$$

$$r = 1 - \frac{1}{\log_2 L} \times 100 \quad (\%) \quad (2.6.17)$$

Nevertheless, in this case the number of instructions-words required is maximum, thus this does not represent an advantageous solution.

2.6.3 Influence of partitioning on the total number of information bits "M" per instruction-word

Now, we will analyze the partitioning consideration when we take into account the totality of information bits per instruction-word "M".

The additional total number of information bits " q_1 " which are required in the worst case, that is, when the drawing has a certain frequency of occurrence $(1-y)$ of straight lines with a greater number of cells than the partitioning considered has, is given by the expression /See Exp. 2.6.7 above/:

$$q_1 = \frac{L}{2^n} (1-y) Mx \quad (2.6.18)$$

where

$\frac{L}{2^n}$ is the maximal number of instruction-words required to represent the longest straight lines from the number of cells point of view

$(1-y)$ is the frequency of occurrence of those straight lines longer than the partitioning considered

- $M=n+v$ is the number of information bits per instruction-word
- n is the minimal number of bits required to represent the partitioning considered
- x is the total number of straight lines in the drawing
- v is the number of bits used for color coding, instruction code, modifiers, etc.

On the other hand, the total number of information bits per instruction-word saved when a given partitioning is considered is given by the expression:

$$q_2 = (k_0 - n) \times \quad (2.6.19)$$

where:

- k_0 is the minimal total number of information bits per instruction-word required when there is no partitioning /See Exp. 2.6.11a/.

From Exp. 2.6.18 and Exp. 2.6.19 above, we can affirm that

- when: $q_1 > q_2$ then a higher number of bits /and higher number of instruction-words/ are required if any partition is done. In this case it is not convenient to effectuate partitioning
- $q_1 < q_2$ then there is saving, even if more than one instruction-word is required. In this case it is convenient to effectuate partitioning
- $q_1 = q_2$ then there is neither saving nor uselessly used information bits in the instruction-words.

Equating Exp. 2.6.18 and Exp. 2.6.19, we obtain the limit

characteristics for each value of "L" /expressed in number of cells/ and " s_n ". Then we have:

$$q_1 = q_2$$

$$\frac{L}{2^n} (1-y) Mx = (k_0 - n)x$$

Substituting Exp. 2.6.3b and Exp. 2.6.11a in the former expression and verifying we have:

$$y = 1 - \frac{s_n}{M 2^n} \quad (2.6.20)$$

Considering the number of information bits per instruction-word "M" equal to an integer multiple "u" of the number of bits "n" required to represent the number of cells included in the partition considered, we have:

$$M = un \quad (2.6.21)$$

Substituting Exp. 2.6.21 and Exp. 2.6.3b in Exp. 2.6.20 we get the definitive expression for "y" as:

$$y = 1 - \frac{s_n}{u(\log_2 L - s_n) 2^n} \quad (2.6.22)$$

To observe the behavior of "y" obtained in Exp. 2.6.22 for different values of "L" and " s_n ", we will give to the variable "u" two different integer values, namely: $u=4$ and $u=2$.

Expressions for finding the value of "y" when $u=4$ for different values " s_n " are given in Table 2.6.4. In each case there appears also the corresponding expression for finding the number of information bits per instruction-word "M" for different values " s_n ".

From these expressions appearing in Table 2.6.4 have been prepared the Table 2.6.5 for different values for "L" in

Table 2.6.4

Exp. 2.6.22 evaluated for different values of " S_n " when $u=4$

S_n	y	M
1	$1 - \frac{1}{8(\log_2 L - 1)}$	$4(\log_2 L - 1)$
2	$1 - \frac{1}{8(\log_2 L - 2)}$	$4(\log_2 L - 2)$
3	$1 - \frac{3}{32(\log_2 L - 3)}$	$4(\log_2 L - 3)$
4	$1 - \frac{1}{16(\log_2 L - 4)}$	$4(\log_2 L - 4)$
5	$1 - \frac{5}{128(\log_2 L - 5)}$	$4(\log_2 L - 5)$
6	$1 - \frac{3}{128(\log_2 L - 6)}$	$4(\log_2 L - 6)$
7	$1 - \frac{7}{512(\log_2 L - 7)}$	$4(\log_2 L - 7)$
...
$\log_2 L - 1$	$1 - \frac{\log_2 L - 1}{2L}$	4
0	1	$4\log_2 L$

Table 2.6.5

Expressions of Table 2.6.4 for different values of "L" when u=4

S_n	L=16 cells		L=32 cells		L=64 cells		L=128 cells		L=256 cells	
	y	M bits	y	M bits	y	M bits	y	M bits	y	M bits
1	0.959	12	0.969	16	0.975	20	0.979	24	0.982	28
2	0.938	8	0.959	12	0.969	16	0.975	20	0.979	24
3	0.906	4	0.953	8	0.969	12	0.977	16	0.981	20
4	-	-	0.938	4	0.969	8	0.979	12	0.984	16
5	-	-	-	-	0.961	4	0.981	8	0.987	12
6	-	-	-	-	-	-	0.977	4	0.988	8
7	-	-	-	-	-	-	-	-	0.986	4
<hr/>										
0	1	16	1	20	1	24	1	28	1	32
<hr/>										

steps of 2^{k_0} in the range:

$$4 \leq k_0 \leq 8$$

$$\text{i.e. } 16 \leq L \leq 256$$

Fig. 2.26 illustrates graphically the values appearing in Table 2.6.5 when $u=4$ for each different maximal length "L" in number of cells.

In Fig. 2.26 there are shown also, connected with dashed lines, those points belonging to different characteristics requiring the same number of information bits per instruction-word.

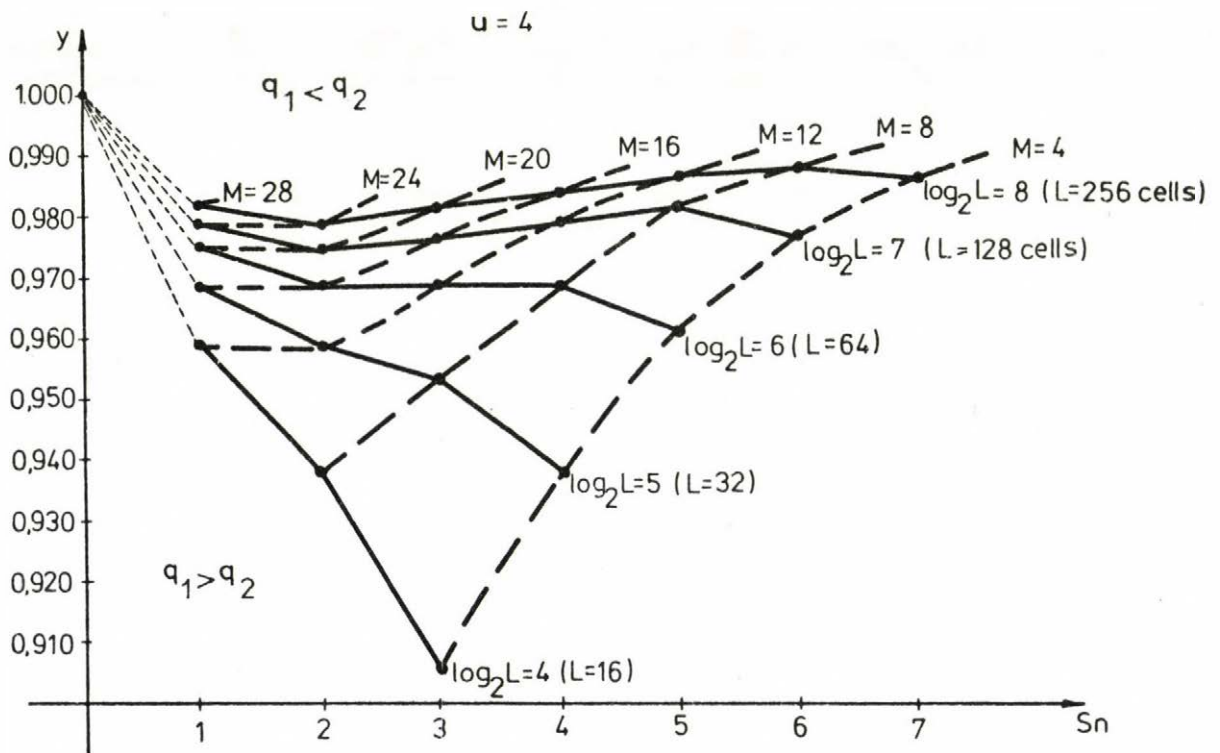


Fig. 2.26

Similarly, expressions for finding the value of "y" and the number of information bits per instruction-word "M", when $u=2$ and for different values of " S_n ", appear in Table 2.6.6.

Table 2.6.6

Exp. 2.6.22 evaluated for different values of " S_n " when $u=2$

S_n	y	M
1	$1 - \frac{1}{4(\log_2 L - 1)}$	$2(\log_2 L - 1)$
2	$1 - \frac{1}{4(\log_2 L - 2)}$	$2(\log_2 L - 2)$
3	$1 - \frac{3}{16(\log_2 L - 3)}$	$2(\log_2 L - 3)$
4	$1 - \frac{1}{8(\log_2 L - 4)}$	$2(\log_2 L - 4)$
5	$1 - \frac{5}{64(\log_2 L - 5)}$	$2(\log_2 L - 5)$
6	$1 - \frac{3}{64(\log_2 L - 6)}$	$2(\log_2 L - 6)$
7	$1 - \frac{7}{256(\log_2 L - 7)}$	$2(\log_2 L - 7)$
...
$\log_2 L - 1$	$1 - \frac{\log_2 L - 1}{L}$	2
0	1	$2\log_2 L$

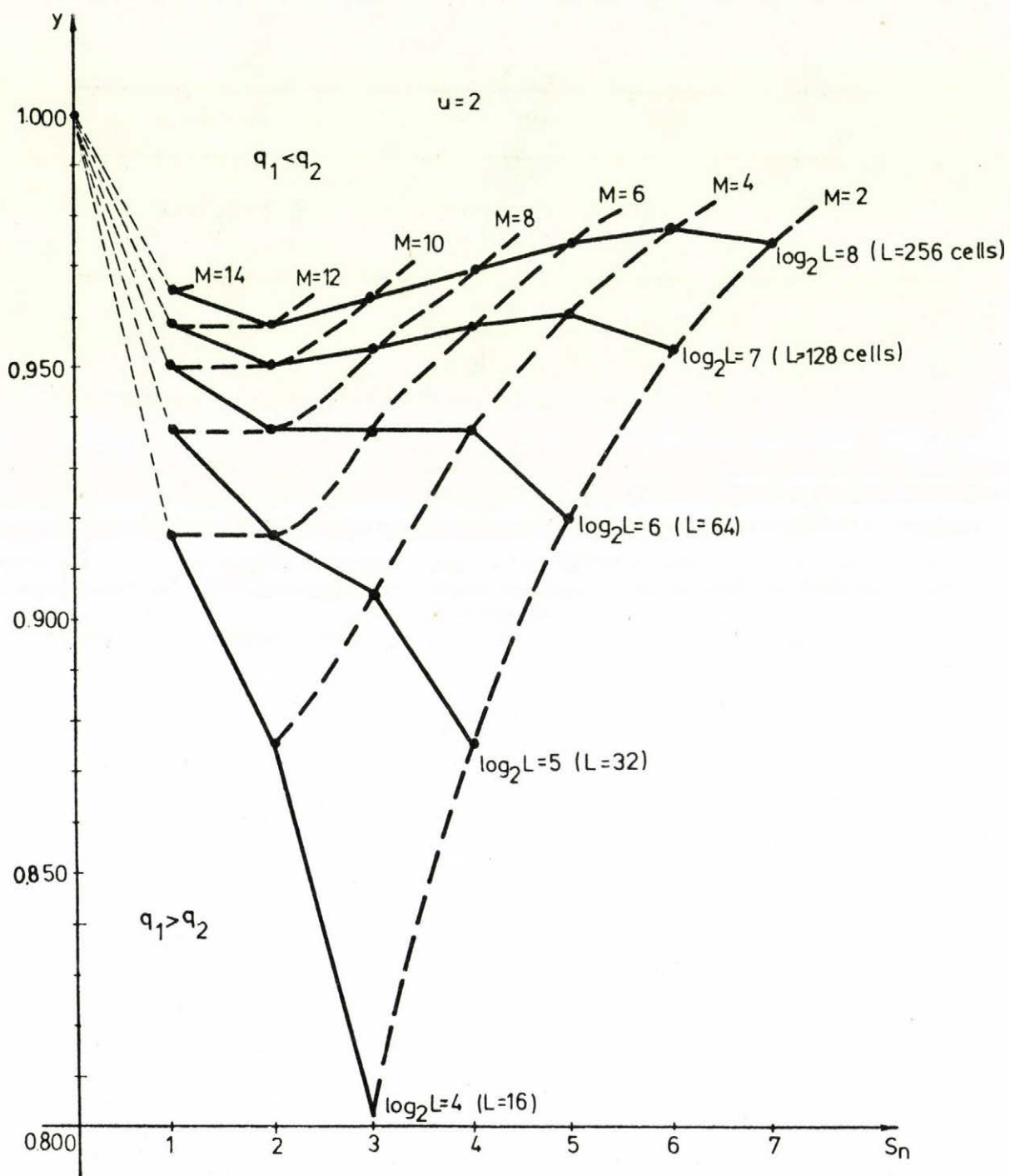


Fig. 2.27

Table 2.6.7

Expressions of Table 2.6.6 evaluated for different lengths "L" when u=2

L=16 cells		L=32 cells		L=64 cells		L=128 cells		L=256 cells	
y	M	y	M	y	M	y	M	y	M
0.917	6	0.938	8	0.950	10	0.958	12	0.965	14
0.875	4	0.917	6	0.938	8	0.950	10	0.958	12
0.802	2	0.906	4	0.938	6	0.953	8	0.963	10
-	-	0.875	2	0.938	4	0.958	6	0.969	8
-	-	-	-	0.920	2	0.960	4	0.974	6
-	-	-	-	-	-	0.953	2	0.977	4
-	-	-	-	-	-	-	-	0.974	2
...				
0.802	2	0.875	2	0.920	2	0.953	2	0.974	
1	8	1	10	1	12	1	14	1	16

Table 2.6.7 includes the numerical values calculated for "y" and "M" from above expressions for different values of "L".

Fig. 2.27 illustrates graphically the values appearing in Table 2.6.6 when $u=2$ for each different maximal length "L" in number of cells. Dashed lines connect those points belonging to different characteristics requiring the same number of information bits per instruction-word.

From characteristics in Fig. 2.26 and Fig. 2.27 we can conclude the following:

1. When the value "u" diminishes that is, when the number of repetition bits "n" represents a greater part of the total number of information bits "M" in the instruction-word or when for a given value "n" is less than the total number of bits "M", then the fan-shaped arrangement of characteristics for different values of "L" displaces down and they are more opened. For this reason the spacing between two successive curves is larger.
2. For greater values of "L", the required frequency of occurrence for a given value of "u" is higher for getting saving of information bits when partitioning is considered. When $u=4$, the frequency of occurrence "y" must be greater than 0.906 for all different values of "L" greater than 16 cells. On the other hand, when $u=2$, "y" must be greater than 0.802 for all different values of "L" greater than 16 cells.
3. For all different values of "u", the lesser influence over the frequency of occurrence "y" for any value " S_n " takes place when the larger length in the useful surface "L" is 64 cells. In all cases, the frequency of occurrence "y" in the range for " S_n ":

$$2 \leq S_n \leq 4$$

is constant in value. This takes place always when the

number of bits per instruction-word is in the range:

$$2u \leq M \leq 4u$$

for every value of "u".

4. For values of "L" greater than or equal to 128 cells, the minimal frequency of occurrence required to get some saving of bits either when $u=2$ or $u=4$, takes place when $S_n=2$. This complements that established by Criterion 1. For $L=64$ cells, this minimal value remains the same in the range for S_n :

$$2 \leq S_n \leq 4$$

5. The value obtained for "y" when $u=4$ and $L=64$ cells, that is, $y=0.969$, is nearly equal but less than the frequency of occurrence of straight lines with length $l \leq \frac{L}{4}$ /See Table 2.6.3/. This means that, with a little security margin, the maximum recommendable value for "u" in order to get saving of information bits when partitioning is considered must be equal to $u=4$, thus Exp. 2.6.21 which gives the maximal number of bits per instruction-word "M" for a given "n" will be then:

$$M = 4n$$

From all these, we establish the following new Criterion:

Criterion 2

When partitioning of straight lines is considered in order to get saving in the total number of information bits required per program and taking into account the previous Criterion 1, the maximal possible length "L" in drawings which renders more advantages related with this will be $L=64$ cells. The maximal number of information bits per instruction-word "M" to fulfil this condition in our application is given by $M=4n$, thus assuring a relative maximum number of

bits for the instruction code, color and other modifiers.

2.6.4 Influence on the aspect ratio of the useful surface "S"

Since all the previous analyses have been carried out considering the longest possible straight line length "L", as the length of that straight line either horizontal or vertical which needs the greatest number of the corresponding symbols arranged successively in the same direction, then all the results obtained are independent of the selected aspect ratio for the useful surface "S" and of the dimensions of the rectangular cell.

2.6.5 Possible saving of bits permitted by the accuracy of drawings

According to Criterion 1 and Criterion 2 established above /Sect. 2.6.1 and Sect. 2.6.3/, best results in this application are obtained when the maximal number of horizontal and/or vertical cells in the useful surface "S" is 64 cells, and when all possible straight lines are represented by means of shorter straight lines having $\frac{L}{4}$ cells /i.e. with 16 cells/ as maximum. In this case, only 4 information bits per instruction-word are needed to represent the corresponding repetitions of the special symbols. These results have been obtained taking into account the frequency of occurrence of straight lines with different lengths required and the total number of bits per instruction-word, in order to give the best use to the number of repetition bits selected and use no more information bits per program than those required when partitioning is not considered.

Now then, based on general characteristics numbers 5,6 and 7 of drawings required for the application /Sect. 2.5/, a new

saving in information bits used for the repetition can be performed.

To carry out this new saving, we have established the following requirements:

1. For a given number of repetition bits "n'" it must make use of the maximum number of possibilities, that is, $2^{n'}$.
2. It must be possible to draw the shortest possible straight line in order to permit a minimal separation, when required, between two components /or other two parallel straight lines/ when we deal with drawings having a relative high density of visual information in some parts.
3. It must be possible to draw the largest partitioned straight line, that is, having $A=16$ cells. This will permit us to draw those straight lines with length greater than 16 cells and less than 64 cells using as maximum 4 instruction-words.
4. It must offer flexibility to programmers when composing the drawings.
5. As far as possible, it must permit programmers to save on useful surface area.
6. It must make possible an aesthetic configuration of the whole drawing.
7. As far as possible, it must not influence the chosen aspect ratio of the useful surface "S" and the selection of the dimensions of the rectangular cell.
8. It must make possible, adding the minimal number of basic short straight segments with lengths less than $A=16$ cells, to draw the maximal number of different length straight lines if required.

These requirements imply the following:

There are only three alternatives to save information bits.

when the selected number of repetition bits "n'" is four, namely, selecting $n'=1$; $n'=2$ and $n'=3$ information bits.

According to requirement number 1 above, the maximum number of possibilities for each one of these values of "n'" are 2, 4 and 8, respectively.

According to requirement 7 above the value $n'=1$ must be neglected, because for aspect ratio of the useful surface "S" equal to or greater than 2:1 or with rectangular cells with very different horizontal and vertical side lengths, there exist limitations to compose the drawings in order to get aesthetic results.

According to requirements 4 and 6, value $n'=3$ fulfills better /than $n'=2$ / the condition for getting good aesthetic configurations and offers more flexibility to programmers.

Now, the selection of such 8 /i.e. 2^3 / basic lengths for the straight lines follows.

According to requirement 2 above, length equal to 1 cell must be included. It can be solved using a single symbol with zero repetition.

According to requirement 3, lengths equal to $A=16$ cells must be considered.

Other values of lengths are selected taking into account requirements 5 and 8.

Let "A" be the maximum possible number of cells which can be represented with the minimal number of straight line instruction-words. According to requirements 8 previously established, we must get with a minimal number of additions /i.e. with only one/ if required, all other "basic" straight

lines /those with lengths less than A cells/. Then, it is necessary that at least one of two segments, the larger one, have a length equal to $\frac{A}{2}$ cells.

Let "B" be the possible lengths, expressed in an integer number of cells, of those straight segments with lengths less than $\frac{A}{2}$ cells.

Therefore, excluding the trivial value of length B=0 cells, the following inequality can be written:

$$A \geq \frac{A}{2} + B > \frac{A}{2} \quad (2.6.23)$$

Subtracting $\frac{A}{2}$ from Exp. 2.6.23 we have:

$$0 < B \leq \frac{A}{2} \quad (2.6.24)$$

With $n'=3$ information bits, there are as maximum $2^{n'}=2^3=8$ different possible lengths for short segments. So far, three values have already been selected, namely:

$$\begin{aligned} B_1 &= 1 \text{ cell} && \text{/requirement 2/} \\ B_2 &= A \text{ cells} && \text{/requirement 3/} \\ B_3 &= \frac{A}{2} \text{ cells} && \text{/requirement 8/} \end{aligned}$$

Taking this into account, Exp. 2.6.24 becomes now:

$$1 < B < \frac{A}{2} \quad (2.6.25)$$

Now then:

- requirement 5 implies the use as far as possible of short straight lines
- due to general characteristics of drawings required in this application, shorter segments are more frequent than longer ones
- with shorter segments it is always possible to compose

longer ones, but this is not possible contrariwise.

Then, 4 of 5 values /instead of 3, or 2/ are obviously chosen in the range specified by Exp. 2.6.25. Therefore, since $A=16$ cells and $\frac{A}{2} = 8$ cells, those values could be:

$$\begin{aligned} B_4 &= 2 \text{ cells} \\ B_5 &= 3 \text{ cells} \\ B_6 &= 4 \text{ cells} \\ B_7 &= 6 \text{ cells} \end{aligned}$$

Table 2.6.8

Values of lengths proposed when $n'=3$

Type of straight line	Basic values "B" /in number of cells/	Other lengths obtained when they are added to basic values selected:		
		16 cells /1 word/	32 cells /2 words/	48 cells /3 words/
1	1	17	33	49
2	2	18	34	50
3	3	19	35	51
4	4	20	36	52
5	6	22	38	54
6	8	24	40	56
7	12	28	44	60
8	16	32	48	64

Finally, the last value, logically, must be chosen equally spaced between the values $\frac{A}{2}$ cells and A cells. Thus:

$$B_8 = \frac{A}{2} + \frac{A - \frac{A}{2}}{2} = \frac{3}{4} A$$

Thus, since $A=16$ cells, then: $B_8=12$ cells.

Basic values "B" finally selected are shown in Table 2.6.8 together with lengths obtained when these values are added to 1, 2 or 3 words with the maximal number of repetitions /i.e. 16/.

2.7 Selection of the number of dots per cell

In reference [74, pp.27], the most common approaches used for generating characters and symbols were commented from the point of view of their principle of work. From this, the dot-pattern character generator was proposed for this application.

Good legibility of characters and symbols depends on a relatively high number of factors /some of them mentioned in reference [74]/, among which the number of dots to have in the dot-pattern matrix undoubtedly is one of the most important. In turn, it depends in the first place on the particular application which in this case is well defined.

According to many studies carried out comparing several rectangular matrixes having different numbers of dots [281], the 5 dots-per-row and 7 dots-per-column dot-pattern matrix presented good legibility to represent the alphanumeric characters and editing symbols. Matrixes with better performance need more dots per row and more dots per column, which for a given screen size presents worse character-per-row and character-per-column relations [258]. Since at the same time

the total number of horizontal dots imposes the frequency of the ROM used, and the total number of vertical dots for a given total number of cells in the useful surface is limited by the number of horizontal scan lines /525 or 625 lines per field according to the TV system used/, then to determine the total number of dots to have in the cell the question consists in achieving the correct tradeoffs.

From the fact that the 5x7-dot matrix is enough to get good character legibility [281] , it has been selected here to represent the alphanumeric characters and editing symbols required by the application. Symbols oriented to the application /in general more complex in shape/, require a matrix /Symbol matrix/ with more dots per row and more dots per column in order to render good legibility.

In order to ensure a relative consistence among the diversity of different integral parts of a picture and a simpler hardware, the 5x7-dots matrix for alphanumeric characters will be considered completely included within the dot matrix proposed to represent the symbols oriented to the application.

The analysis carried out in this study will be initially based on geometric factors in order to get a good performance, i.e., to ensure a good legibility, a good aesthetic quality, an optimal efficiency to represent all symbols required and the fulfilment of all requirements imposed by the application. Thus, the result of this analysis is considered as a first approach.

Firstly, the number of horizontal dots per row "m" for the symbol matrix will be analyzed.

From that stated above it can be written

$$m \geq 5$$

(2.7.1.)

Taking into account that characters and editing symbols require at least either a free leading or a free rear column of dots to ensure a minimal separation between successive characters in a string of letters or numerals, then Exp. 2.7.1 must be written as:

$$m > 5$$

The upper limit for "m", as it was previously pointed out, determines the frequency at which the information must be read from the ROM for a given number of cells per row.

Taking into account criteria for the Symbol-set numbers 3 and 8 /Sect. 2.5.2/, in order to ensure that those symbols which are symmetrical with a vertical axis crossing through the geometrical center of the matrix have always an optimal coupling with upper and lower symbols adjacent to them, then an odd number of horizontal dots is required.

Thus, "m" may acquire the values:

$$m = 2s - 1 \quad (2.7.2)$$

where "s" is an integer number in the range:

$$4 \leq s \leq \infty$$

This helps to create optimally the Symbol-set proposed for limited graphics and render good aesthetics for pictures.

From values obtained from Exp. 2.7.2, one of the first two values for "m" /i.e. 7 or 9/ must be selected. The value $m=11$ will give 6 columns ($11-5$) between adjacent cells which is more than a "space" character between every character, and this is not acceptable in editing.

If $m=9$, then there are 4 columns between each character having 5 columns each. This means that if in texts, tables, etc., "c" represents the number of characters per row, then

the total number of horizontal dots " D_h " required will be

$$D_h = 5c + 4c$$

considering that the string of letters begins with a character and finishes with an intercharacter spacing.

Therefore, the efficiency " $e(\%)$ " of this type of matrix in editing is:

$$e(\%) = \frac{5c}{4c+5c} \times 100$$

$$e(\%) = 55\%$$

which is not a high value. Thus in texts, 45 % of the total number of horizontal dots used has been lost in intercharacter spaces.

If $m=7$, the efficiency will be:

$$e(\%) = \frac{5c}{2c+5c} \times 100$$

$$e(\%) = 71.4 \%$$

which is more acceptable. Now, only 28.6 % of the total number of horizontal dots is used for intercharacter spacing.

Concluding, the number of horizontal dots per row " m " proposed for the symbol's matrix in the first instance is 7 dots.

This value imposes a horizontal frequency which can be easily handled by present electronic technologies when a comprehensive number of cells per row is selected.

In similar form, the number of vertical dots per column " n " for the symbol matrix is selected. From that stated above:

$$n \geq 7 \quad (2.7.3)$$

Taking into account that characters and editing symbols require at least either a free upper or lower row of dots to ensure a minimal separation between successive rows of characters in tables, texts, etc., Exp. 2.7.3 must be written as:

$$n > 7$$

The value for "n" is limited, among other reasons, by the number of useful scans per field /a number less than 525 or 625 scans per field depending on the TV system used/ and the desired maximal number of cells per column to have in the cell-organized useful surface. At the same time, once the number of horizontal dots "m" has been selected, fulfilment of criterion for the Symbol-set number 7 /See Sect. 2.5.2/ limits also in some measure the number of vertical dots "n". Additionally, since actual separation between successive scan lines varies with the screen size, the TV system used, the type of phosphor, the diameter of the electron beam, and with other factors, then the definitive value for "n" is influenced also by the aspect ratio desired for the actual cell.

Taking into account criteria numbers 3 and 8 /Sect. 2.5.2/, in order to assure that those symbols symmetrical with a horizontal axis crossing through the geometrical center of the matrix have an optimal coupling in every situation with left-hand side and right-hand side symbols adjacent to them, then an odd number of vertical dots is required.

Thus, "n" may acquire the values:

$$n = 2r - 1$$

where "r" is an integer number in the range

$$5 \leq r < \infty$$

The lower the number of vertical dots, the higher the

possible number of cells per column; moreover, since on the one hand the distance between successive horizontal scan lines is normally greater than the hypothetical distance on the screen between successive horizontal dots into the scans themselves, and on the other hand the maximal number of scan lines to compose the useful surface can always be reduced to the number required, then 9 scan lines will look visually like more than 9 scan dots; thus, the value for "n" must be the lesser permissible one. Therefore, the value $n=9$ is selected in first instance as the maximal one possible.

If the number of useful scan lines is supposed as 480 lines /e.g. from the 525 lines per field of the american TV system/, then the maximal number of cells per column can be $\frac{480}{9} = 53$ cells, which are really enough to select the must appropriate number for getting a nice aspect ratio for the pictures with 64 horizontal cells per row /or maybe a little more for convenience/.

Therefore, in the first instance, the symbol matrix proposed to represent the symbolic information within a cell has 7 dots per row and 9 dots per column.

As it was pointed out at the beginning of this analysis carried out above, this proposition can not be considered as the definitive solution for this application. The reason for this is the fact that the final solution can be given only after the hardware implementation in breadboard is realized, taking also into account /in so far as when the 5x7 character matrix was selected/ the related ergonomic requirements.

Among other possible alternatives which could fulfil somehow the requirements imposed /apart from the 7x9 matrix/, are the 7x8-, the 8x9- and the 8x8-dot matrixes. They, in some measure, give a representation of independent symbols with

a little less aesthetic appearance, but render equivalent results for the whole picture.

Each one presents some advantages with respect to the 7x9 matrix, although some disadvantages as well, e.g. the 8x8 matrix permits the representation of continuous or intermittent slanted straight lines at 45° and 135° with respect to the horizontal, which cannot be attained with the 7x9 dot matrix /nor with the 7x8 or 8x9 dot matrixes/ using only one class of symbol; the 7x8 dot matrix renders the possibility of more cells per column than the 7x9- and 8x9-dot matrixes and more cells per row than the 8x8 and 8x9 ones, and so on.

With these matrixes the intercharacter separation is of 2 or 3 columns and the interrow separation is of 1 or 2 rows. All these values are admissible from the aesthetical point of view, render good performance and fulfil the basic ergonomic requirements.

To justify the above conclusions, Fig. 2.28 shows three cases where equivalent solutions with respect to the whole picture are achieved for three different types of symbol matrixes /or what is the same, sizes of cells/. Fig. 2.28a shows the four possible cases to couple a horizontal straight line with a vertical straight line, or viceversa, using a straight segment /required for instance in the complex symbol of a tank/; Fig. 2.28b shows horizontal and vertical single symbols of a diode /or rectifier in general/. Fig. 2.28c shows an alternative for the complex symbol of instruments /in this case a fluxmeter/.

Obviously, the graphical items included within the cells are not always completely the same /as in Fig. 2.28a/, but they will never have unacceptable differences.

Considering the results obtained from examples in Fig. 2.28,

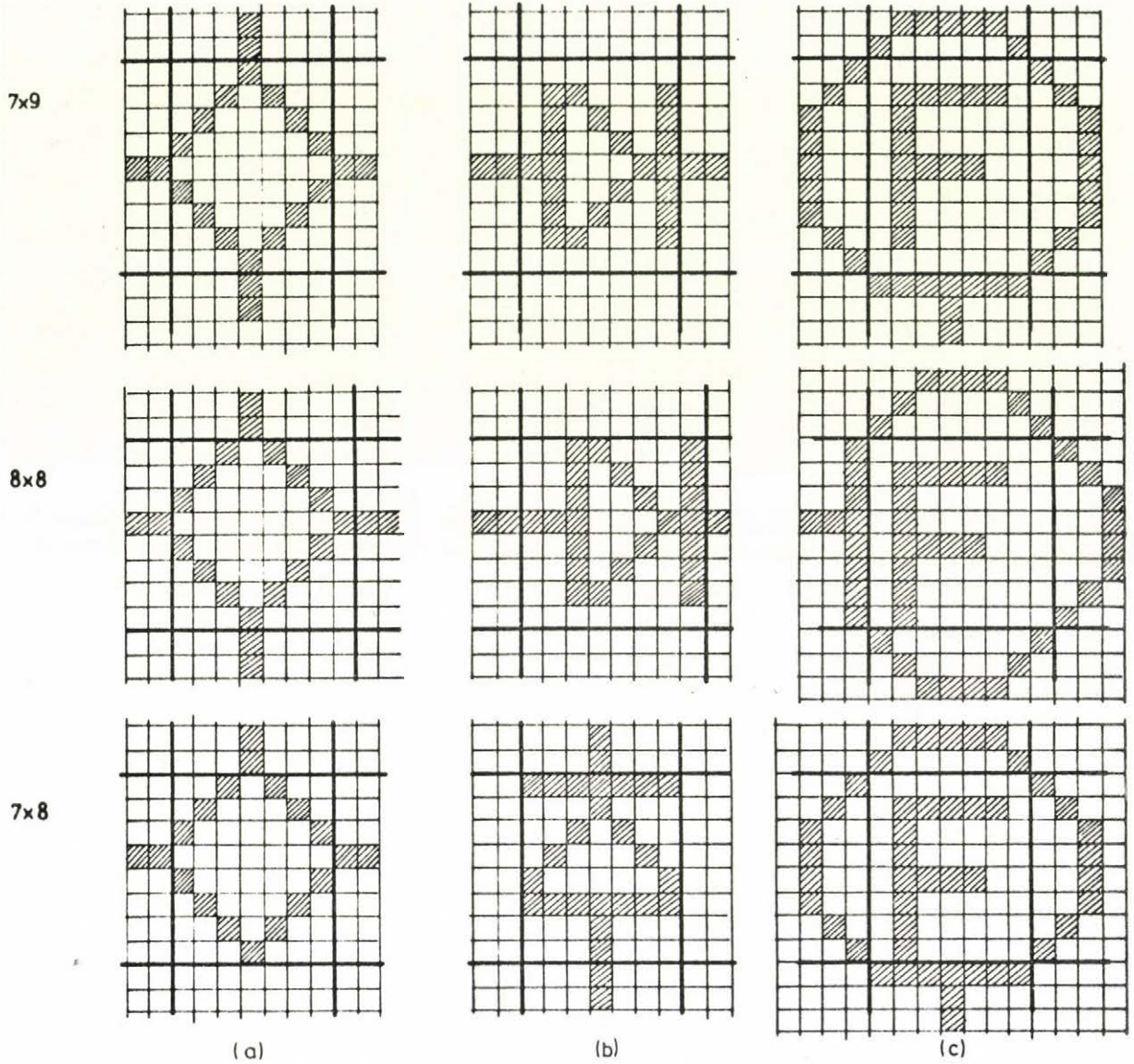


Fig.2.28

there now exist more degrees of freedom to select the most appropriate dot-matrix type for this application.

To carry out the best selection, the analysis from the geometrical point of view of those most important symbols required by this application is necessary.

Intrinsic characteristics of symbols used in this application have been enumerated in Sect. 2.5.2. According to this, some modifiers are proposed for single symbols in order to get several new symbols from each one stored in the read-only memory /ROM/ used. With this alternative, the ROM is used more efficiently, a greater diversity of symbols is achieved using the same codes plus modifiers for the same number of symbols and finally, higher accuracy can be obtained in creating vertical bar diagrams /See Sect.2.5.3.4/.

In order to attain the best and more efficient use of information bits required for the modifiers, it will be seen that the most convenient number of raster scans per row of symbols must be eight. Thus, the use of 8x8 or 7x8 dot-matrixes is more advantageous. From these two matrixes, the 8x8 dot-matrix is selected in order to fulfil criterion number 7 for the Symbol-set, to make possible slanted straight lines either continuous or intermittent at 45° and 135° degrees with respect to the horizontal and to get better separation between successive characters in a row.

The modifiers can be obeyed by hardware in the display itself in a very straightforward form and with a very simple logic.

Therefore, the 8x8 dot-matrix is proposed for this application based on all the considerations pointed out above.

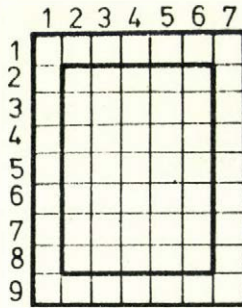
The efficiency "e(%)" of this type of matrix in editing with respect to the number of characters located in a row is:

$$e(\%) = \frac{5c}{3c+5c} \times 100$$

$$e(\%) = 68.5 \%$$

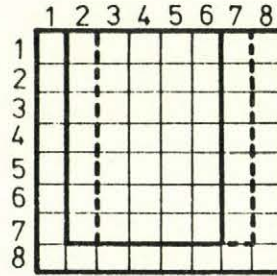
which can be considered as an acceptable value.

Alternatives for the location of the 5x7 character matrix inside the symbol matrixes proposed are shown in Fig. 2.29.



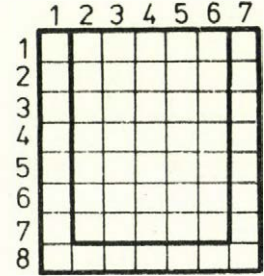
7x9 matrix

(a)



8x8 matrix

(b)



7x8 matrix

(c)

Fig. 2.29

These solutions are based on the following reasons:

- In order to permit writing of letters or numerals aside single symbols with a convenient separation in all directions, the 5x7-dot character matrix is located in the innermost part of the symbol matrix. In case of 7x8- and 8x8-dot symbol matrixes in which this is not possible, it is suspended in the uppermost border, thus application oriented symbols which do not fill wholly the symbol matrix, must always be located in the lower part of the character matrix to get similar results.
- The location of the character matrix within the 8x8-dot symbol matrix is shown to the left /with continuous line/ to be consistent with examples shown in Fig. 2.28 for the same symbol matrix. Another possible and geometrically equivalent solution is shown with dashed lines in the figure.

CHAPTER 3

THE DISPLAY AS A COMPONENT OF A PROCESS SUPERVISION SYSTEM

3.1 Introduction

The first part of this Chapter deals with the analysis of some general characteristics observed in components of an industrial process. Based on these characteristics, in the second part the fundamentals of a Manipulating System for their supervision and control have been proposed, mainly related to the graphical display units to be used and the application-oriented computer operating system which will perform the tasks required by the whole Supervision System.

We start from the fact that we have an industrial plant whose complete process should be automatically monitored, supervised and eventually controlled with help of a digital computer and those auxiliary equipments required to carry out automatically the collection of information from the process, the transmission of commands to the process, the continuous real-time centrally-located supervision performed by trained operators, the preparation of production reports and in general, all the activities necessary to permit the process to work automatically, with high reliability and with minimal operator participation.

In their most simple form, these tasks could be sketched as shown in Fig. 3.1. In this, common link lines between the industrial process being supervised and the computing system used for this task are shown.

The industrial plant, from the instrumentation point of view, might be divided in several different sections. Each

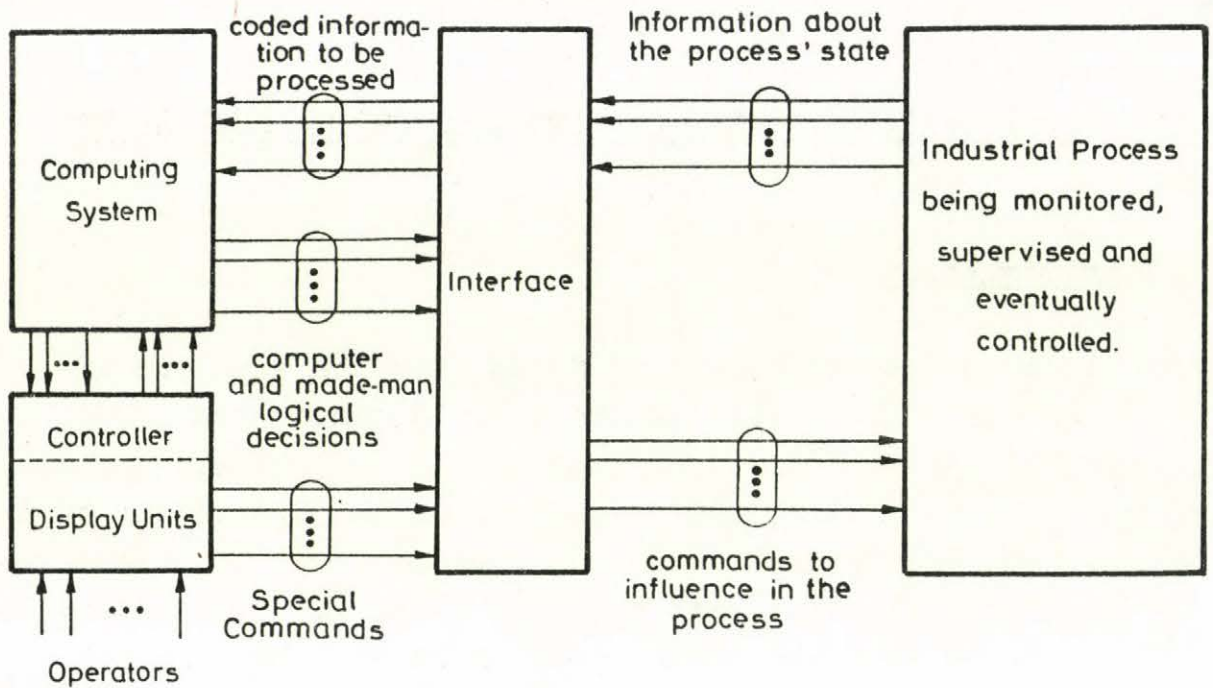


Fig. 3.1

section should have instrumentation of a different type, electromechanical equipment, measuring devices, hydraulic, pneumatic, electrical or other types of components, etc., altogether adequately interconnected to perform the corresponding tasks for which they were included in the process. There can also be some integral parts in the plant which are not dynamic from the operation point of view /i.e. pipe lines, tanks, etc./, but because each of them has an actual state each moment /full tank, empty tank, active or inactive pipe line, and so on/, they will also be considered as components.

3.2 Definition of some terms used in this Chapter

Component:

We define as a component, each element of the industrial process which is

periodically supervised or monitored, and when required, also controlled. It can be either a measuring instrument, a pump, a pipe line, or a complete plant requested for information about its production, operating conditions, etc.

Component Vector: Consists of a given number of computer words wherein all further required data about each component are orderly specified.

Control System: The system which together with the Supervision System makes it possible to control specific components in the process. This control is commanded by computer decisions or by operator commands.

Identifier-word: Is the coded distribution of bits sent by every component of the process, which permit us to identify the origin of the "state" information arriving together with it, each time a component is interrogated by the computing system.

Pointer List: A computer list wherein the addresses /or pointers/ of all the Component Vectors are orderly stored. They are addressed directly by means of the Identifier-word arising from the process.

Section:
/or Control
Section/ Each arbitrary group of components within the complete plant, factory, etc. /normally interconnected in the process for performing a given operation/, and which can be isolated and detailly sketched into a symbolic di-

agram to facilitate supervision and control by a trained remote operator. Examples of sections in a process can be a complete pump station /or part of it/, or the vapor generating plant, or the pneumatic network used for controlling the system, or a complete factory or production unit when considered as a component within a greater system.

State-word:

Is the coded distribution of bits sent to the computing system by every component of the process which contains all the information about its actual and current state. It is sent always together with the corresponding Identifier-word each time a component is interrogated by the computing system.

Supervision Center:

The place where the computing facilities, control facilities, display consoles, operators, etc. charged with the supervision and control of the whole system are located.

Supervision System:

The Supervision System comprises those means, sensors, computer programs, etc. used for monitoring the complete industrial process.

Symbolic diagram:
/or drawing/

A detailed sketch of a section in the industrial process being supervised and controlled. It can also take the format of a table, list, or any other graphical distribution of data used to show information to operators. In

general, it is composed of symbols of components, straight lines, alphanumeric characters, etc.

Table Alfa:

Consists of a computer table created for the updating of Picture Files. With help of this table, the Picture File Address and the initial address of the associated Picture Modifying Program are searched for.

Table Beta:

Consists of a computer table created for sending the updating information to display units. It stores the display code of each display unit associated with the code of the last Picture File requested by it.

Vector List:

A computer list wherein all the Component Vectors related to all the components are orderly stored on a one to one basis. They are addressed by indirect addressing by means of the Identifier-word arising from the process.

3.3 General characteristics observed in components of an industrial process

From the most general analysis, there can be established the following general characteristics /and given some convenient particular observations in each case according to our objective/ related to the components used in any industrial process, namely:

- Each component /i.e. measuring instruments, electro-mechanical devices, etc./ is physically located in only

one place, invariant with time.

Since all the components have precise and fixed physical locations within the process, a given number of inter-related components can be grouped into different Control Sections. Each Section would have its own symbolic diagram. Superpositions between two or among several Control Sections could be avoided if one of the superposed parts is considered as a new independent Control Section. In this form, the possibility that one or more of the components belongs to more than one drawing does not occur. This is illustrated in Fig. 3.2 with an example.



THREE SECTIONS WITH SUPERPOSITIONS
IN THE SHADED AREAS

THE SAME SECTIONS TRANSFORMED AND
CONSIDERED AS SIX DIFFERENT SECTIONS

Fig. 3.2

When more detailed information is required from some part of a given picture shown on a display screen, in case of sophisticated graphical displays it is commonly extracted using windowing facilities. In our conception, when this is the case, the more detailed part must be considered as a new Section with a new independently programmed symbolic diagram. However, specific components could occasionally appear in more than one symbolic diagram. In these cases, it is necessary to specify some priority level among several different drawings containing a given common component /or components/, in order to select automatically that drawing with the higher priority level when an emergency

situation takes place with the common component to all them. More than two different priority levels are not required, because in alarm situations it is sufficient to show only one diagram of all those containing a common damaged component. Nevertheless, updating of common components must be done by the computer in all the diagrams containing any number of them.

Division of the whole process area in Control Sections depends on the density of components desired within each one of them, that is, on the level of detail desired for each drawing. It does not depend on the physical dimension or area occupied by the components considered, because we deal always with the symbolic representation of the corresponding Control Sections considered in the plant, but of course, it depends on the total area of the available useful surface on the display screen.

- Every component realizes, either continuously, periodically, intermittently or even occasionally, precise and predefined tasks all of them inherent in its particular design and purpose.

Components with only one task and those with more than one task, operating regime, stable states, etc., must be independently considered from the control point of view.

- The action or effects caused by the operation of each component, can influence only one Section, more than one Section, or the whole plant.

However, each Control Section can be always considered from particular points of view as a whole, in spite of the possible direct influences arising from other Sections, because in each Control Section all the variables related to it should be generally measured independently of /although influenced by/ the actual state of the

remainder of the plant. A given hierarchy order must be established in these cases when the whole system is designed.

- Components are commonly of different nature, that is hydraulic, pneumatic, electric, electromechanical, electronic, etc.

Due to this and because the information required from them is sensed by different types of sensors, it may be required that their outputs be opportunely transformed to electrical signals by the corresponding transducer equipment.

- Some types of components supply only analog values about their particular state /i.e. measuring instruments/, or inversely they adopt different stable states for analog values of a variable /i.e. valves with continuous setting possibility, etc./

This means that in some step of the information system, the conversion of analog signals to digital signals and viceversa will be required.

- Some components require control and others do not require it.

In common industrial plants, control operations are performed automatically in the component itself by means of sensors, stabilizer circuits, regulators of different types, etc., in general only based on specific operating conditions of the particular component being controlled. When a highly automated system is implemented and a control is required depending on logical or other decisions given by the computing system or by the operators /to close a valve, to start a pump, etc./, then the control must be performed with the help of a given command sent

opportunately from the Supervision Center. In the first case the control is local and depends mainly on the current component's state; in the second case the control is remote and depends mainly on the conditions of the remainder of the plant, on operator decisions, or on previously conceived /and programmed/ computer decisions. Both types of controls can be used together on the same component, because the local control performs mainly regulation tasks over a given variable previously set for the component, while the second one performs the change in the component of the value of other particular variables related to it. As a general rule, not every component would require this remote control.

- Each type of component is periodically watched by the operators from particular points of view and with different objectives at a given frequency, or what is the same, it is supervised at different time intervals /each second, each two minutes, etc./

The corresponding time interval must be defined previously depending on the type and characteristics of the component, its dynamic stability, etc., but it will be advantageous if the possibility is present of requests by means of commands given by the operators located in the Supervision Center.

- The total number of components of different types required by a complex plant, is high.

This implies special design of the system in connection with the scanning, information processing, searching, and so on.

- The reliability and durability of the components and auxiliary means must be, and is always expected to be, as great as possible.

When new means with given specific purposes are added to a process, as a general rule they must have higher reliability and longer durability than the existing parts.

- When a new industrial plant is designed, it always uses the most up-to-date components at that time.

At present, modularity and unification must be considered, among others, as basic considerations to be taken into account in new designs.

3.4 Basic information required about the process to be supervised and controlled

The preliminary analysis of the process to be supervised and controlled must offers, in particular, information about:

1. Total number of components which are going to be supervised in the process and how many of them are going also to be controlled automatically by the Supervision System or by some specific commands given manually by the operator who supervises it. A condition for the system is that all supervised components do not necessarily require to be controlled, either automatically or manually, but all components to be controlled must be periodically monitored. When determining this number we must consider a security margin for possible extensions or modifications of the process lines.
2. Minimal and maximal possible frequency of repetition at which the components could be requested to inform the Supervision System about their actual state. It is also necessary to know the exact frequency between these limit values at which each component should be independently requested.

3. Minimal number of information bits required for encoding all the necessary information about the actual state of the more "exigent" component. When determining this number, some few additional bits for possible future requirements of the Supervision System must be considered.
4. Different types of components, possible data which should be requested from them, their controllability, their location in the process, total number of components of each type and any other specific information about the objective of the whole Supervision System.

3.5 Most important coded information required to and from each component in the process

From each component to be supervised within the process, it is necessary for the computing system to receive when required, at least, coded information about:

- Actual stable state and/or actual operating regime and/or physical state /in maintainance, in normal operation, etc./ and so on, of components. In the particular case of sensors, the measured value with a predetermined precision is required.
- Its identity /i.e. the type of component, a specific order number within the corresponding type or within the total number of components, etc./ for the subsequent analysis, processing and other operations, using the "state" information arising simultaneously with it. This information must be enough to the computer in order to know with what component it is dealing, for locating the results of analysis in the corresponding files, for determining the position information within the corresponding graphical symbolic diagram /or diagrams/, etc. The identification might also be carried out starting from the scanning position data offered by the scanning system used.

- A signal to cause an interrupt in the computing system via an Interface, to prepare the conditions for getting the state information and the identification data sent by the requested component. This signal could be an answering signal to the synchronizing pulse provided by the scanner used for selecting each component.

Towards each supervised and possibly controlled component within the process, it is necessary to send from the computing system when required, at least:

- A request signal inquiring from the component about its actual stable state, its actual operating regime, its physical state, last measured value, or any other data necessary to the computing system. This signal can also be that used to identify the requested component.
- The corresponding control signals for actuating the component /i.e. to connect or disconnect it, to change the setting, etc./
- A synchronization signal to synchronize the Supervision and Control system with the process being supervised and controlled.

3.6 Manipulating System. Generalities

In Reference [74] we have analyzed the alternatives in connection with interactive graphical displays when they are used in process control applications.

The analysis establishes that when a large number of displays are used in a Supervision and Control system /which is the case in a process control application/, the best alternative for the category selection is when this task is performed by the computer. Additionally, it recommends the refresh memory to be included in the display's hardware

either when computers of limited memory capacity are used, or in case of multi-display systems, or when long communication paths are used. It is observed also that in case a magnetic disk is used as back-up memory of the computer system, and data transfer between the disk memory and each display unit can be carried out without the interruption of the CPU operation of the computer, then refreshing of displays could be accomplished by the disk.

Basing some arguments on these criteria, we can point out the following:

- The display's refresh memory capacity is limited to a maximal number of m -bit words given mainly by the number of cells in which the rectangular useful surface of the screen has been hypothetically subdivided.

The total number of cells "Z" is given by:

$$Z = M \times N$$

where:

M is the number of cells per row and

N is the number of cells per column.

The refresh memory is considered as an object-oriented memory having a "cell map" with a total number "Z" of cells [154].

Analyses of the values for "M" and "N" and of the total number of bits per word "m", have been carried out in another part of this study /See Sect. 2.6.3/.

- The information stored in any display's refresh memory must be only that previously requested from the computer. It belongs to only one drawing of the "M₀" possible

drawings previously stored and continuously updated by the computer. The selection of the corresponding drawing /category selection/ must be performed by software by the computer. Updating of information must also be performed only by the computer which continuously receives the actual information through the Interface from the process being supervised. The updating information related to a particular drawing "i" must only be transmitted to the particular display unit showing the drawing "i" at that moment, but only in the case when it is different to the information previously stored in the computer memory, and therefore, to that currently stored in the corresponding refresh memory of the display. Otherwise, this transmission must not be performed. The decision about the occasion when the updating information should be transmitted to the display units, might be implemented either by hardware or by software. If it is done by software, it must be performed by program in the computer; if it is done by hardware, this decision must be taken by the Controller.

It is not advisable for state information to arrive at the Display Controller directly from the process /but rather at the computer Interface/ because of the following reasons:

- If it is sent directly from the process, /considering the best case when it can be interpreted directly by the display units/, then it would be necessary to get also together with the state information the position information of the component in the corresponding drawing /or drawings/ and also the code of the drawing /or drawings/ to which it belongs. This is particularly complicated when the system deals with a great number of components and many sections, areas, etc. as in this application. Thus, from a given information provided by the components together with their "state" information, the computer should determine which type of component it is dealing

with, it selects which drawing /or drawings/ it belongs to, it finds out the particular position of the component inside the selected drawing, etc. Only after this, will the computer could update the previously stored information with the new one recently received from the process and effectuate the prescribed calculations.

- After receiving all the information about the component /identification code and state information/, the computer would perform some calculations in order to detect abnormalities, to find out overall values, etc. As a product of these calculations also, the computer should decide, among other things, the change in color /if any/ to have in the symbol of the previously analyzed component, data, etc. Obviously, if the updating information is sent to the Controller directly from the process, these calculations can not be performed before they are shown onto the display's screen. This does not exclude the possibility of a display with processing facilities, but it would make more expensive the cost per unit and would impose new requirements for the whole system. The use of firmware, microprocessors, and other technologies in this field, is beginning just now promising to be the solution for the future, although at present they do not substitute all the facilities of a minicomputer yet.
- If we consider that the only task of each display unit consists in showing to operators as a "robot" the information previously handled and formatted by the computer for every cell, then we have that:
 - auxiliary memory circuits are less complex because of the sequential order imposed
 - it needs less capacity, because of it is not necessary to store the corresponding position information belonging to every character, symbol, etc. to be shown on the screen. This makes possible a

considerable saving of memory in the display

- Display refresh memory can be refreshed always in the same form, i.e. following always the same sequence of cells from the first upto the last one
- the overall cost of display units is considerably reduced.

Concluding, the information arising from the components of the process being supervised, must arrive first to the computer through the Interface and there, after the analysis of the "state" information, the determination of the particular drawing /or drawings/ to which it belongs together with the exact positioning address inside it /or them/, and after other tasks are carried out, it must be sent /through the Controller/ to the corresponding display unit showing in that moment the drawing including the component which has been supervised. How all these tasks can be carried out is dealt with later in this paper /See Sect. 3.6.3.3/.

Starting from the fact that the display's memory capacity is limited to a given number of m-bit words corresponding each one to each cell, there are two very different forms to store in the computer memory the information related to a drawing and to transmit it from the computer to the display units, namely: in prefixed order and without order.

If the information /updated or not/ must be sent to the display units in order, it implies that in the computer memory there must be stored an exact /updated or virgin/ replica from all points of view of the information to be transmitted to a given display, i.e. of the picture file. In general, this is required in some form by the system, but in this case, the graphical coded information related to a particular drawing must each time be transmitted sequentially cell by cell and in a predefined order to the display unit which previously requested it. With this alternative, the

display does not need to receive position information /because it can be internally generated/, but the system presents the following disadvantages:

- the computer requires, each time an updating takes place in a given drawing being shown, to transmit cell by cell all the stored graphical information related to it;
- the computer must leave free locations for those cells having no information;
- those advantages offered by data compression in the computer are lost.

If the information can be sent to the display units without order /at random/, it implies that a position instruction must always precede the graphical instructions. This alternative offers the advantage that the computer can save memory locations /because it does not need to leave free locations when some cells have no information/, but it could have three main disadvantages, namely:

- updating is more complicated, because of the difficulty met in searching for the exact location in the picture file wherein the information related to a particular component has been stored; this can be overcome by means of computer tables created for this purpose and adequate algorithms to carry out this task;
- it requires /at least/ a number of computer words equal to the double of the total number of graphical instructions belonging to each drawing in order to store, preceding each one, the respective position information related to it; this can be overcome if the position information is calculated each time an updating takes place, it then being unnecessary to store all of them, which would require in complex pictures a relatively high additional number of

locations in the computer memory;

- it could also present serious difficulties when the system requires enlargements or modifications and as a product of this, the total number of locations destined to each drawing in the computer /size of the picture files/ is less than the maximal possible capacity of information required to show a drawing on the display screen; this can be overcome either making the picture files sufficiently large /which is not so economical from the point of view of memory capacity required/ or creating conditions for getting a considerable data-compression for storing the information related to every drawing.

From the analysis carried out above, it is seen that the alternative of storing the graphical information at random in the computer memory and transmission at random of the visual information to the display units, would present more advantages than if these task are performed in sequential order. All that is necessary is to create the conditions which will permit us to make the whole system realizable.

Firstly for this, some further design criteria are proposed for the display units.

3.6.1 Display Unit

3.6.1.1 Three design criteria for the display units

The following three criteria are established and discussed for the display units:

1. Each display unit must be able to interpret directly, completely by hardware and without processing, all the specially created instructions arriving to it.

A good solution is attained when these instructions are

only of two types:

- position instructions /not stored explicitly in the display's refresh memory/ required unconditionally mainly when updating tasks are carried out over some part of the drawing being shown;
- graphical instructions /stored in the display's refresh memory/ required to define the different symbols, numerals, etc. which compose the drawings. They include also the color and other modifiers related to each type of graphical element.

These instructions can be easily obeyed in a similar form as an alphanumerical display does.

Now referring to characteristics numbers 2 and 3 of drawings required in process control applications /See Sect. 2.5/, some compression of data is attained when information related to horizontal and vertical straight lines of limited length is represented digitally coded /See Sect. 2.6.1/. Due to particularities of the solution proposed here in order to get a given data compression /a single phenomenon of repetition/, the instructions related could be easily processed /or broken up/ either by software in the computer, or by hardware in the display unit or in the Controller.

The alternative of doing it in each display unit is neglected because:

- it will complicate the hardware design,
- it would be convenient and advantageous that the behavior of the display be the same for all types of graphical instructions,
- it will increment the cost per display unit,
- it will obstruct the data transmission when some graphical instructions must be broken up and others do

not require it.

If it is done by hardware in the Controller, similar inconveniences are present than those present when they are broken up in the display units, namely:

- the hardware design of the Controller will be more complex,
- the size of the buffer required in the Controller will be large in order to handle favorably the diversity of instructions to be broken up before being sent to displays units,
- it will increment the overall cost of the whole system.

However, this solution present one significant advantage:

- it saves appreciably the computer time in relieving it of this routine work, there being thus longer available time for more serious processing.

If it is done by software in the computer, the inconveniences enumerated above are not present, but the computer will loose time for performing these routine works. However, in this case, it is possible to get a high constant transmission rate and straightforward breaking up of the compressed instructions.

Now, instructions must be stored in compressed form in the computer memory, either if the breaking up of compressed instructions is done in the computer by software or in the Controller by hardware or firmware. In such a way, computer memory capacity can be saved, it being advantageous that all the calculations, decisions, updating, etc., be done over the coded computer word of a single compressed instruction and not independently cell by cell. Moreover, compressed instructions must be

broken up only when they are transmitted to the display units; in any other situation they must be kept in compressed form. It is proposed also to create direct access for I/O operations between the computer memory and the refresh memory of the display units, in order to facilitate possible transmissions without obstructing the normal computer's processing.

On the other hand, some complex symbols such as symbols for pumps, valves, tanks, etc., or large squares, rectangles, etc. which require two or more cells for their graphical representations /using several single symbols with their own meaning/, can be created with graphical subroutines. The reference position of these types of subroutines could be referred or not to a particular cell belonging to the complex symbol. Graphical subroutines are stored in the computer's memory, being handled by the specially created routine called up by the respective graphical instruction. Color code and other modifiers, are generally specified in each independent graphical instruction composing the graphical subroutines. These instructions can also in their turn be any type of compressed graphical instructions, meaning that a multiple level subroutine's handling facility will be convenient for the computer used in the system.

Moreover, if we consider that graphical subroutines be in some form relocatable, i.e. that they can be called up by every picture file created when they are needed by them, and because these graphical subroutines could be independently called when required by means of only one graphical instruction belonging to the graphical language created, then a considerable data compression could also be attained, thus further saving computer memory locations. For this reason and because of the frequency with which graphical subroutines are commonly required, they

could be considered as a high level data compression means of the whole system.

Finally, it is advantageous that graphical subroutines be broken up in single graphical instructions /maybe including another type of compression of data/, only in those cases when they must be transmitted to display units.

Hitherto, handling of program subroutines has been performed by software in common computer systems, having widely proved its effectivity in many other applications. For this reason among others, it could be advisable here also as the best alternative, to perform in the computer by software the breaking up of all graphical subroutines conceived for the system. If this criterion is extended, breaking up of those instruction-words which present data compression based on the repetition of equal symbols /which is carried out more easily, requiring less time and in a more straightforward form/, would have an economical justification if it is done by software also in the computer.

2. The graphical instructions which produce the drawings on the display's screens must specify all the information related to a single cell of the screen's useful surface.

This information will be constituted mainly by the instruction-code, color-code, symbol identifier and modifiers considered for each type of graphical instruction. In this form, in order the display be able to store all the information related to any one of the total number of drawings created for the whole system, and considering that is no possibility for superposition of symbols according to criterion number 6 for the Symbol-set /See Sect. 2.5.2/, then it is sufficient to provide a display refresh memory with constant capacity determined by the

total number of cells in which the screen has been hypothetically subdivided, and by the number of information bits-per-word considered for the display's graphical instructions /See Sect. 2.6.3/. Thus, it will have only an independent m-bit memory word destined for every single cell of the screen. Refreshing sequence will be always the same and performed orderly, independently of the type of information stored in the refresh memory of the display. The sequential order of refreshing the information to be shown on the display's screen is such that it permits the display's instructions to be successively called up internally, provided that the location containing the information of each cell is arranged row by row and column by column according to the path of the electron beam travelling along the screen. In this form, the refresh operation can be easily made orderly by the display unit itself /similarly as in alphanumerical displays/ without the necessity of using associated position information to do it. Then, this task can be carried out by hardware independently in each display unit, not requiring computer time for this.

The situation is different with respect to the updating of the graphical information shown on the screens of display units.

In this case, each updating display instruction supplied by the computer must be sent to the displays together with the corresponding position instruction and with the code of the display unit which shows the particular diagram including it. Position instructions arrive at the displays also from the computer via the Controller. All this information must be sent by the computer only if at least one display unit is showing the picture wherein the component related is an integrating part. If any display unit is showing the drawing to which the updating in-

formation belongs, then no transmission of information must take place from the computer to the Controller. Nevertheless, the computer picture files are updated in every case. Graphical display instructions with their corresponding position instructions and display code, are received from the computer based on the state-words arising from the monitored components of the plant and on the computer lists prepared and stored in the computer operative /or back up/ memory. Synchronization of each display unit can be handled by the Controller itself in order to send to each one the updating information received each time from the computer, thus relieving it of this task.

3. Display units must cause interrupts in the computer only when a new particular drawing is requested by operator command, or when a control command is given by the operators to the system through the display unit itself.

This means that in any other cases the computer can not be interrupted by the display units. This permits the computer to process a longer time the information arising periodically from the components of the process.

Inclusion in the system of new drawings created in order to enlarge it, once the whole system is functioning correctly, must be carried out separately and by responsible people only. This could be done sharing the computer resources to avoid serious problems in the real-time operation of the system. This imposes a new requirement to the computer machine and operating system to be used.

All information to be sent to the display units, must be transmitted through specially designed I/O channels which do not obstruct the normal operation of the computer, thus ensuring truly real-time operation for the whole system.

As a result of three criteria pointed out above, the flow of the information when a specially dedicated display unit or any other input-output peripheral equipment is used to create pictures, can be considered as that shown schematically in Fig. 3.3.

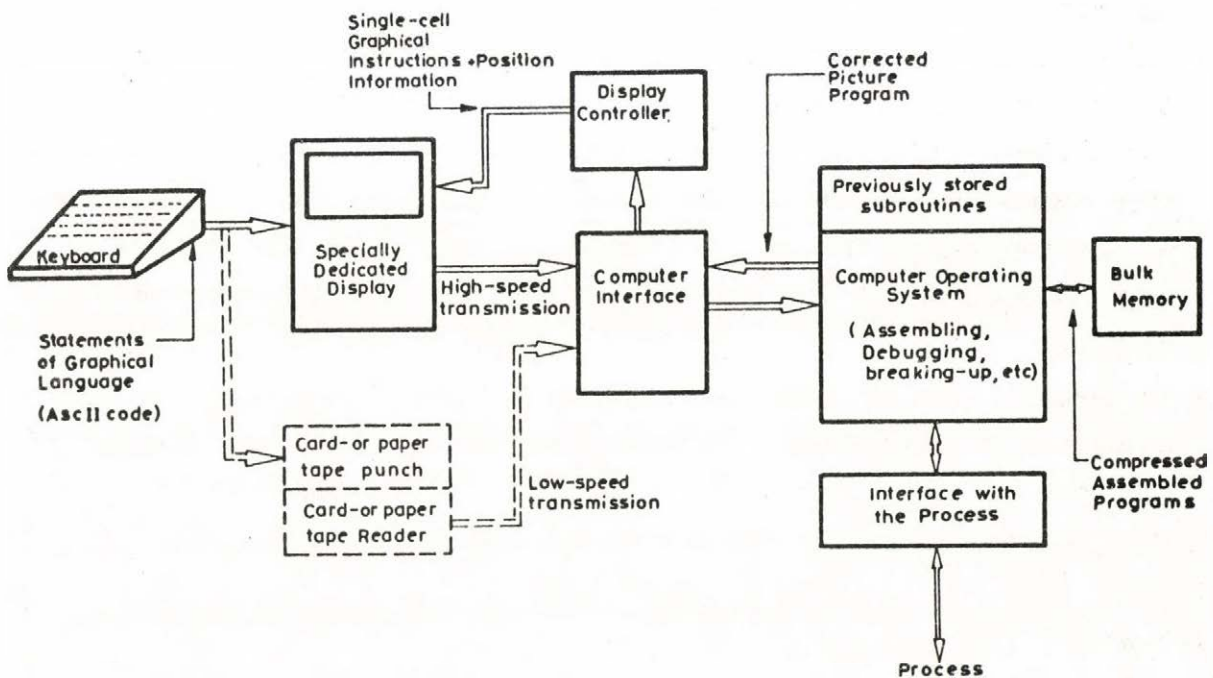


Fig. 3.3

3.6.2 Computer Operating System

3.6.2.1 Tasks to be performed by the computer system with the information from the process

It is now supposed that unambiguous information about the actual state of a particular component /the "State-word"/ and information about its procedence /the "Identifier-word"/ arrives simultaneously at the Interface.

By means of this coded information from each component, the computer system, in the most general form, must carry out the following tasks:

- to identify the component which sends it,
- to determine to which drawing it belongs,
- to determine the correct position of the symbol which represents the component within the previously selected drawing,
- to select the program /or programs/ which handles the information received for updating the corresponding file /or files/ related to the drawing /or drawings/ to which the component belongs. In case the updated file refers to a drawing being shown by some display unit, the system must also update the refresh memory of the corresponding display,
- to select the program /or programs/ which handles the dynamics of drawings, in order to effectuate the visual changes within them as a product of those changes which have taken place in the state of the component.

3.6.2.1.1 Identification of the component

The information for the identification of the component must arrive at the computer from each supervised component in the "Identifier-word" arising from the process. The total number of bits "m'" of the "Identifier-word" depends either on:

- the total number of components "N" to be supervised,
- the format given to it, which depend in turn either on the number of components "N", or on the total number of drawings prepared for the system and the number of components per drawing, or on the number of different types of components "T" considered in the system and the number of components of each type, etc. In every case, a given number of spare bits to be used in possible future expan-

sions or other purposes must be considered.

The particular selection of the most appropriate format should depend on the specific characteristics of the industrial plant to be supervised, although the total number of bits to have in the Identifier-word is nearly the same for all cases, because it depends finally on the total number of components "N" to be supervised in the plant.

Types of Identifier-word formats

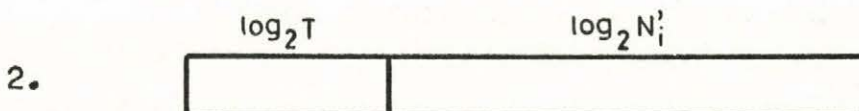
- Since the physical location of each component in the plant is unique, the code used with this format can be easily associated to each one, thus facilitating maintenance.
- It is possible to use it as an addressing-word for the computer memory. In this case indirect addressing is required in order to get previously stored information about the characteristics, type, etc. of the component sending its actual "state" information.
- It permits us to verify more easily the validity of the Identifier-word received.
- If some expansion takes place in the system, it is not necessary to modify substantially the programs, lists, tables, etc. In this case, what is required is to complete the lists, tables and so on, with new data corresponding to each component added to the system.
- It permits us to use consecutively the locations of the computer memory destined to store all data relative to the components /in their corresponding Component Vector/ of all drawing of the system. Thus, a relative saving of computer locations can be obtained, because it is not necessary to leave free spaces for the information related to components not existing yet.
- It does not require us to store the information related to each component sequentially into the computer memory for each independent drawing or for the whole system if a Random Access Memory /RAM/ is used. Thus, it permits us to carry out easily the modifications /enlarging/ of the lists, tables, etc. when expansions in the plant take place.
- It permits us to use a constant-length code for all the

components in the system.

- Since a RAM can be used, the time required for fetching all data related to any component in the system is always the same. Besides this, it requires a relatively easy algorithm, the same for all the components.
- In case a component belongs to more than one drawing, it can be indicated by means of only one bit in the Component Vector. In this case, all data related to a given common component which refer to every drawing of which it is a part, must be stored one after the other in the computer memory.

Disadvantages of this alternative are:

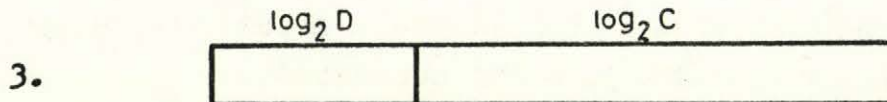
- It requires a higher number of bits per component in the computer memory to define all the data which identify it /type of component, order number within each type, etc./, because the Identifier-word is not used as data, but only as identifier.
- It uses a little more computer time per component due to the indirect addressing required to find all the information related to the component, i.e. for searching the corresponding Component Vector.



where: " $\log_2 T$ " is the logarithm to the base 2 of the number of different types of components "T" in the system or, if the logarithm is fractional, the next larger integer.

" $\log_2 N'_1$ " is the logarithm to the base 2 of the num-

ber of components "N'" of the type "i" in the system or, if the logarithm is fractional, the next larger integer.



where: " $\log_2 D$ " is the logarithm to the base 2 of the number of drawings "D" in the system or, if the logarithm is fractional, the next larger integer.

" $\log_2 C$ " is the logarithm to the base 2 of the number of hypothetical cells "C" into which the display screen has been subdivided. It gives the position of the symbol in the particular drawing " D_i " of the system. If the logarithm is fractional, the next larger integer gives the total number of bits.

4. Any other combination of data relative to the components which permits their identification.

In all these cases, the Identifier-word must be considered as data, it can not be considered as address due to the possible discontinuity in the ascendent order of the code.

Advantages of alternatives 2., 3. and 4. are:

- They need a lower additional number of bits per component in the computer memory to define all data which identify it, because they are themselves data-words.
- They could permit us, at first glance, to have directly some information either about the component's type, or about the diagram to which it belongs, or about the position of its symbolic representation in the drawing, etc.

depending on the code used.

Disadvantages of these alternatives are:

- The Identifier-word requires more bits, due to the practical impossibility of using exactly all the 2^T , or 2^D , or 2^N , etc. different codes in each case, for "T" different types of components, "D" different drawings, "N" different components in the drawing, etc. This is a consequence of the variable-length code required in these cases complicating considerably the processing in the computer, the algorithms required, etc.
- Determination of the number of spare bits for possible future expansions in the system is difficult, since it may be necessary to change the whole system /lists, programs, etc./ if a relatively large expansion takes place.
- They make it more difficult to implement the whole system due to the existence sometimes of useless bits in the Identifier-word.
- They require relatively much computer time for fetching in a given Table stored in the computer memory, all other data corresponding to the component which sends the Identifier-word. This time will be different for each component depending on its position within the Table.
- In case one component belongs to more than one drawing, it would then be necessary to refer it somehow to two or more different Component Vector in a Table, or send two or more Identifier-words to the computer when the component informs it about its state. These solutions will complicate considerably the processing and the format of the Table.
- The verification of the validity of the Identifier-word

received is more difficult, there must be a List in the computer equally ordered than the arriving order of Identifier-words at the computer system.

From the previous analysis, we conclude that from these given alternatives the most advantageous is the first one, that is, that which consider the Identifier-word arising from the component as a fixed-length addressing-word for the computer. Nevertheless, we do not affirm categorically that this one is the most convenient solution in every case, although it seems to be the most commendable in large systems.

Using this alternative, there follows a form by which all data referring to all drawings can be stored in the computer memory, in order to process the "state" information arising from the components together with it for updating the picture files. At the same time, we will continue the analysis of each computer task enumerated in Sect. 3.6.2.1, in order to give consistency to our criteria.

3.6.2.1.2 Component Vector

About each supervised and already identified component of the process it is necessary to know:

- Drawing to which it belongs
- Type of format of the arriving state-word in order to interpret it
- Coordinates of the cell wherein the symbol representing it appears in the drawing
- Type of Element with which the computer will deal.

These data compose the Component Vector of each component in the process.

In case some component be a part of more than one drawing, these data logically must be specified for each drawing

independently. They preferably must be stored one after the other in the computer memory, thus requiring an easier algorithm for updating. This eventual possibility can be indicated with only one bit in the last word of each Component Vector.

The number of computer words required to specify wholly the corresponding Component Vector, is determined by the size of the system and the characteristics of the components which compose it. They are addressed by indirect addressing using the respective Identifier-word received from the components.

This is schematically illustrated in Fig. 3.4

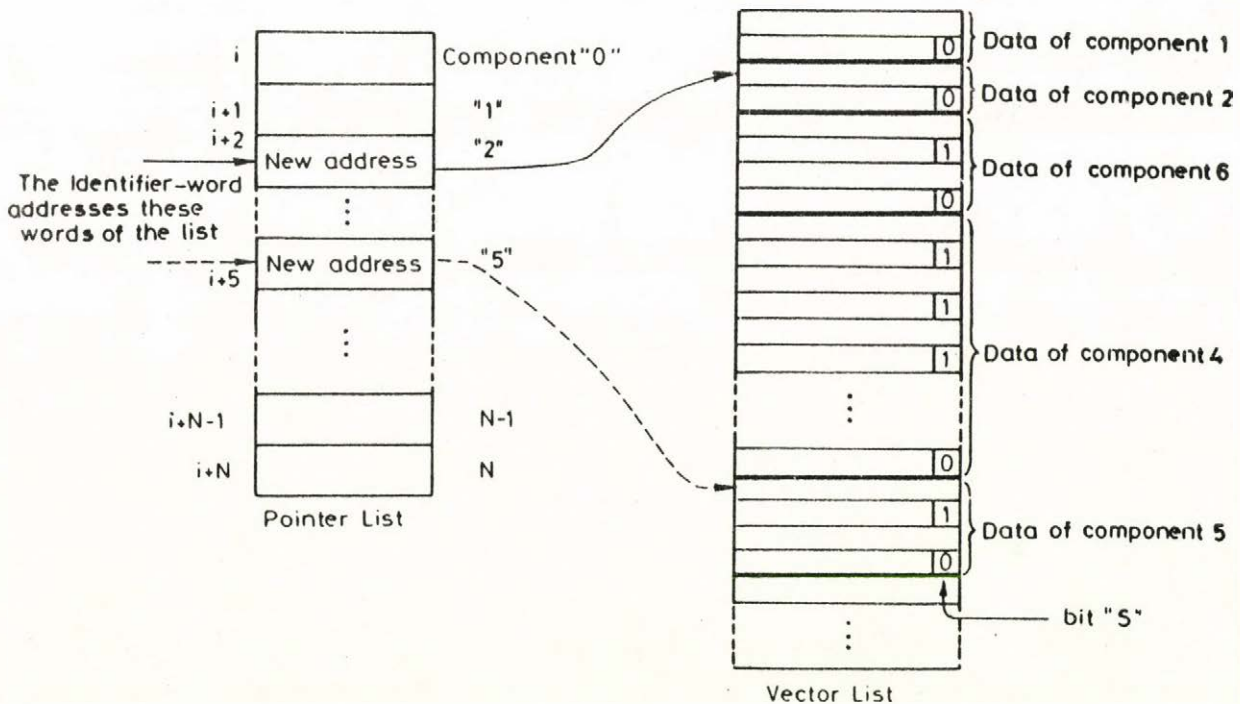


Fig. 3.4

In Fig. 3.4, bit "S" has a value "0" if the subsequent data-word group is related to another component; it has value "1" if the succeeding data-word group is related to the same component /that previously addressed/. Therefore, the last data-word group related to the same component in the sequence /with an arbitrary number of them/ must have the bit "S" with value "0".

All these data are previously stored in the computer memory according to the specific drawings prepared for the system and to the conclusions of the further analyses in this paper.

Addresses of all the first words of all the Component Vectors of the Vector List /having or not several successive data-groups/ related to one component, are stored successively in a Pointer List, either orderly or without order. To each Component Vector there corresponds one address in the Pointer List on a one to one basis, although a component may have an arbitrary number of data-word groups within the Component Vector assigned to it. Words of this List are addressed by the Identifier-word arising from the components in the process.

Data included in the Component Vector

Each data-group of the Component Vector related to each component of the whole system must include the following necessary data:

1. Code of Picture

This code is automatically created by the Assembler from the Name given to the picture by the programmer.

It is used to find by means of the Table Alfa, two addresses closely related to each component, namely:

- the Picture File Address to which it belongs,

- and the address of the Picture Modifying Program /See Sect. 3.6.3.2/ associated with it.

2. Type of Format

Formats of state-words may be mainly of the following two different types:

- Discrete code

In this case the state-word is considered as composed by a given number of independent bits each one having a predefined meaning, similarly as a microprogrammed instruction /e.g. bit number 3 with value "0" could mean that the component is out of service; thus bit 3 with value "1" means then that the component is in operation, and so on/.

- Numerical code

In this case the state-word is considered as a numerical value. In turn, we may distinguish two types:

- Type 1: those numerical values which are visually represented on the picture and therefore necessarily stored on the Picture File
- Type 2: those numerical values which are not visually represented on the picture, but nevertheless must be stored on the Picture File.

Nevertheless, this does not exclude the possibility of using some other type of numerical code required by the system.

3. Displacement /See also Sect. 3.6.3/

The displacement represents an always positive value relative to the initial address of the Virgin Program, i.e. to the Picture File Address. The number of bits used to represent the value of this parameter must be sufficiently high to permit the access to any location

of the Picture File. Thus, on the one hand, any location of the Picture File could be easily addressed by Indirect Relative Addressing with a computer register as basis. On the other hand, the contents of the first location of the Picture File is the absolute coordinate values of the cell to which the remainder of the cells of the whole picture are relatively related. This contents is specified in the Absolute Position Statement which compulsorily constitutes the first statement of every Virgin Program.

Adding algebraically to this contents the relative coordinate changes /if any/ caused by each statement, starting from the second upto the previous one to that related, the corresponding cell coordinate values related to each statement can be calculated.

4. Type of Component /or Type of Element/

This datum is required only if there are two or more discrete code formats for state-words. If a common format is given to all state-words having discrete code independently of the possible different types of components, then this datum is not required. Otherwise, each different kind of discrete code will require a different routine to handle it.

If the state-word consists of a numerical code, then this datum will be meaningless.

3.6.3 Picture File

In general, only two programs of very different nature are related to each picture or drawing, namely:

- a Virgin Program
- a Picture Modifying Program

3.6.3.1 Virgin Program

The Virgin Program /VP/ is the first program to be created for each picture. Its only objective is to define the picture topologically using the different types of statements of the Graphical Language specially created for this application /See Chapter 4/. Depending on the efficiency of the Graphical Language and on the skill of the programmer, it can be prepared in a more or less compressed form.

VP can be loaded at the computer using the keyboard of a specially dedicated display or using any other input peripheral /paper tape reader, card reader, etc./ During this, keyboard and display /or console/ work together like an alphanumeric display when used as input peripheral device in a common computer system.

In writing VPs through the console, the color used to display the alphanumeric characters composing the syntax of statements, must be immaterial.

Each VP must be identified with a "Name" following the letter "V" /Virgin/ in this form

V NAME

This command shall permit us to determine the type of program being loaded and the place wherein it must be stored in the computer memory.

The VP constitutes the beginning /i.e. the first part/ of the file created for each picture.

Virgin Program can be considered as having two intimately interrelated parts: one fixed, always invariable for each picture, and another variable according to the actual conditions of the plant or process and to the current operating state of the components symbolically represented in the

pictures. Statements composing the fixed part of each particular drawing do not ever change; on the other hand, those of the variable part must be updated in real time by the computer system, according to the information received from the process in state-words. This updating takes place when the Virgin Program /source program/ has already been transformed to a comprehensible data structure /object program/ with help of the Interpreter created in the system for this purpose.

Every drawing must be particularly designed the first time having all the components included into it in their initial steady-state condition. This graphical program which produces the drawing in its initial steady-state conditions is precisely the Virgin Program of that drawing. It is considered as a program including all the components inside a given section, area, etc. of the plant in the corresponding initial steady-state /if any/, i.e. before they are put in operation.

In general, if any area or section is modified in some measure or is enlarged with new components, in these cases the Virgin Program /and possibly the programs which handle it/ must be accordingly modified.

Preparation of Virgin Programs is done by software using the specific particular instructions included in the special-purpose Instruction-set of the Graphical Language created for this particular application /See later Sect. 4.4/.

Every Virgin Program of a drawing will have:

- a particular code /expressed by a name/ which unambiguously identifies it. This code identifies also the drawing produced by it and the whole section, area, zone, etc. which it represents, independently of the actual state of components which compose it;

- an absolute position instruction to which the remainder of instructions of the whole Virgin Program are referred to. This instruction must be the first one in each Virgin Program and permitted to appear only once within it;
- an invariable number of instructions representing graphically the initial steady-state of all the components which compose it. This number is indirectly limited by the maximal number of cells in which the display screen has hypothetically been subdivided. Because of the characteristic number 11 of drawings /See Sect. 2.5/ and the data-compression obtained with the Instruction-set created, the total number of instructions in a Virgin Program will be generally fewer than the total number of cells in the useful surface of the display;
- an instruction indicating the end of the Virgin Program.

3.6.3.2 Picture Modifying Program

The Picture Modifying Program /PMP/ is the second program related to each picture. It is programmed using the different types of statements, instructions or commands belonging to any of the General-Purpose Computer Programming Languages for which the computer has a Compiler, or to any of the Assembly Languages prepared for it.

The PMP can also be loaded at the computer through the keyboard of the specially dedicated display like the VP, or through any other means /card reader, paper tape reader, etc./

In writing PMPs through the console, the color used to display the alphanumeric characters composing the statements must also be immaterial.

The PMP can be identified with a "Name" preceded by the

letter "M" /Modifier/ in this form:

M NAME

This command shall permit us to determine by computer routine the type of program being loaded and the particular Virgin Programs to which it is associated.

The Picture Modifying Program states all prescribed operations required to update the corresponding VPs to which it is associated. Each time a state-word arising from the process is received at the computer system, it is analyzed in real time by the appropriate programs prepared. These programs compose the System's Supervisor Program which verify the validity of the state- and identifier-words received from the plant, detect emergency situations, compare measured values with their corresponding permissible limit values, etc. They can be considered the fundamental programs of the whole system, since they handles all types of information received from the plant and interrelate them. Once the results of this analysis are obtained, they are used for updating the picture files related to them by means of the corresponding PMPs. Then, the PMP, designed taking into account the particular dynamics of the part of the plant represented in the drawing, modifies the whole picture resulting from previous changes carried out within it, by using the values stored at the end of the Picture File /"Z" Values/ received from the System's Supervisor Program.

A particular PMP may be associated to one or more VPs. This is caused by the normal interrelation among different pictures. A change in the state of a given component, may cause a change not only in the picture to which the particular component belongs, but in general in many pictures related to it /e.g. subsequent pictures, tables, comparative diagrams, and so on/.

This implies that in general the number of PMPs is less than

the number of VPs, so a corresponding Table is necessary which associates VP names to PMP names. Although in complex plants this solution would not be the most convenient, a common PMP could occasionally be associated to all VPs created for the system.

As a result of this, which is the common case, PMPs are not stored in picture files, but in another part of the computer's operative memory.

3.6.3.3 Structure of the Picture File

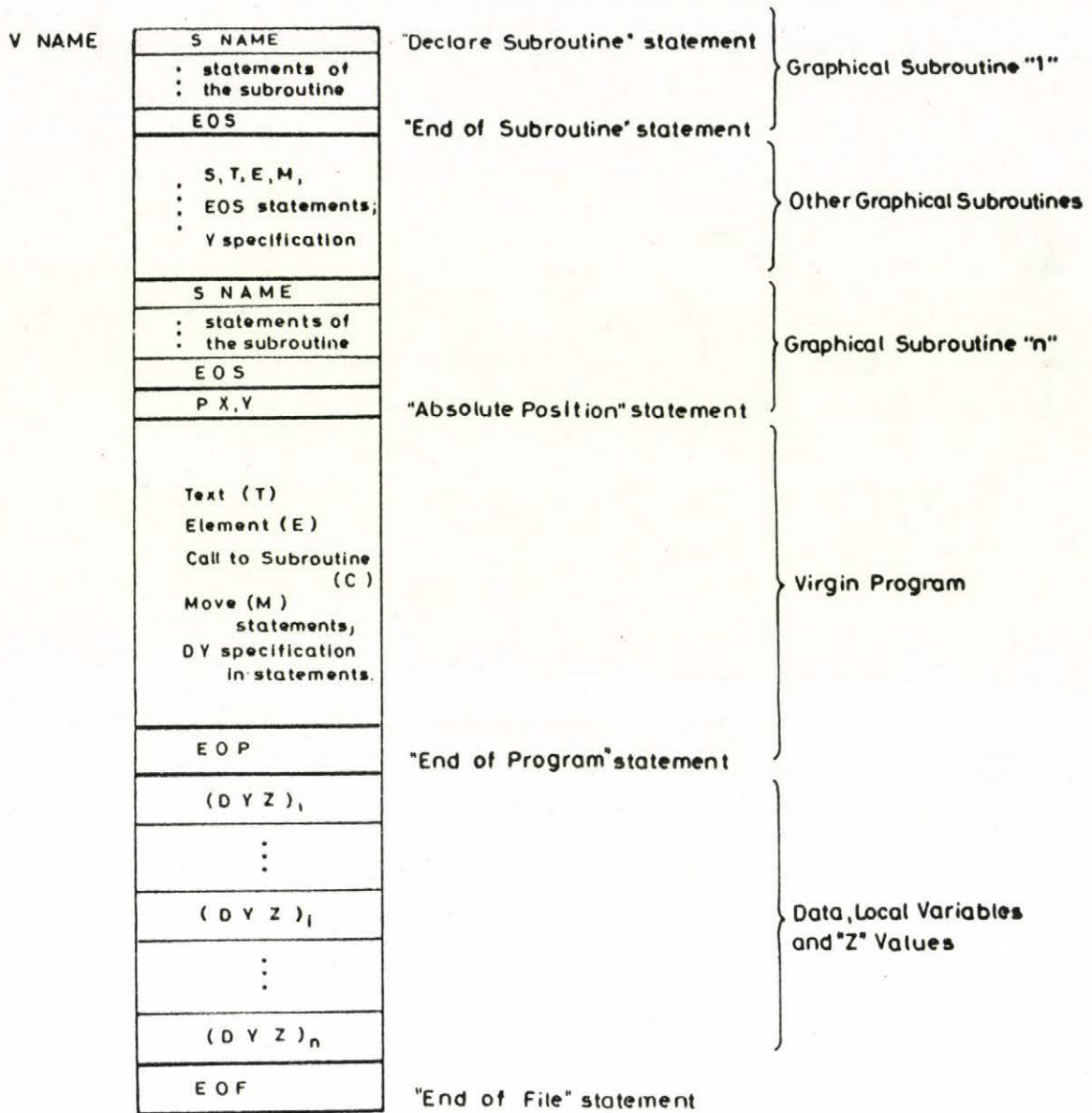
The structure of Picture Files is shown schematically in Fig. 3.5.

It is composed by the following parts:

- An arbitrary number of graphical subroutines

These graphical subroutines are those required to create the particular Virgin Program of the Picture File. Another alternative could be to create a stock of relocatable subroutines to be used by all the Picture Files of the system. In this case, they are stored altogether in a particular file in the backup computer memory accessible to all the Virgin Programs created for the whole system. In this form a greater data compression is obtained with the Graphical Language specially created to build-up the Virgin Programs, since then they do not need to be declared in each Picture File each time they are required.

In general, graphical subroutines are composed by a "Declare Subroutine" statement, a given number of Graphical and Move statements and by an "End of Subroutine" statement. The number of graphical subroutines in a Picture File is unlimited /See Sect. 4.4.2/.



NOTE:

(DYZ)_i means Data, Local Variables and "Z" Values of the "i" Call to Subroutine statement.

STRUCTURE OF THE PICTURE FILE

Fig. 3.5

- A Virgin Program

It is the most important part of the Picture File. It defines topologically the drawing created for the section, area, etc. of the plant, or the tables, lists, bar diagrams, etc. used in statistical records, economic

reports, etc.

- An arbitrary number of "Z" Values, Local Variables and Data

The "Z" Values are those values required for updating the Virgin Program stored in the Picture File by the Picture Modifying Program associated to it. The number of "Z" Values depends on the characteristics of the picture, its design, etc. and consist either in digitized data received from the plant /or other factories/, or calculated by the System's Supervisor Program. For Local Variables and Data, see the Sect. 4.4.3.1.

- An "End of File" statement

This statement indicates the end of the file. Starting from this, a new Picture File can be open. The initial address of each Picture File is defined by the "Name" given to the Virgin Program stored in it.

3.6.3.4 Updating of Picture Files /PF/

Updating of PFs can be carried out in at least two different forms depending on the general conception of the whole system. Here two cases are possible:

- state-words arrive at the computer independently of whether there were changes in the state of the components
- state-words arrive at the computer only in those cases when a particular component changes its state.

In the first case, a comparison must be performed between the updated information and the information previously stored in the VP related to it. Thus, the computer will send to the display units and will change the respective computer words of the VP, only in those cases when one differs from the other i.e., when it has really been modified. In any other case, no changes will be carried out.

If the second alternative is used, every state-word carries information. Thus, the updated codes of the updated state-ment can always be sent without comparison to the corresponding picture file and displays, with the security that it will cause changes in the picture being shown.

In any one of the two cases, the goal must be to send to display units only the information which has been recently updated. This must be carried out each time by the Picture Modifying Program.

Additionally to the above when the updating is carried out, we must take into account that between any two successive updatings of a particular drawing, two alternatives can be considered as possible, namely:

- each time to run again the Virgin Program /i.e. compile it completely or maybe only a part of it/ before the updating is performed. In this case, the Picture Modifying Program modifies only computer words of the data structure which are in initial-condition state. This alternative is possible in those cases when all the updating information previously used is stored in some place of the Picture File, or in those drawings /or part of them/ in which each time only one from many alternatives is permitted;
- to update the data structure starting from the current conditions of the drawing after the previous updating was carried out. This alternative would imply not only to modify the computer words which were in initial condition state in that moment, but to modify also those which were previously updated and now require to be changed to the initial condition state.

The first alternative would require a little more computer time when the updating is carried out, but the Picture Modifying Program required in this case will be considerable

simpler.

The second alternative, on the other hand, would complicate somewhat the Picture Modifying Program, because each time the whole data structure must be practically analyzed, thus saving time only with those statements of the Virgin Program which are fixed, i.e. those which never change their particular state.

The first alternative can be carried out in a more straightforward form, although the best solution depends each time on the characteristics of the particular drawing being updated. In order to save computer time, both alternatives could be used provided that the required software is created for this.

Since this study deals mainly with display units and with the computer operating system which should handle them, the sequence of the operations required for picture updating is dealt with below in general form. Because it should depend in great measure on the data acquisition system used to collect the data, on the available computer resources, etc., they are not taken into account in order to give more generality to the present study. However, some considerations are given each time they are closely related to the display units or to the computer system.

3.6.3.4.1 Sequence of operations for picture updating

One of the most important tasks of the computer operating system integrating the whole Supervision and Control System is the updating of picture files. This task must be carried out each time a new state-word arises from the process. Its analysis by a particular computer program will finally cause the complete updating of the data structure of the respective Virgin Program by means of the Picture Modifying Pro-

gram associated to it.

The Updating Program is called up each time a state-word arrives at the computer independently of the picture to which it is related. However, the Picture Modifying Program related to one or more Picture Files, is called up only when the state-word is related to the Virgin Program corresponding to these files.

In other words, the Updating Program involves those common operations which are carried out independently of the type, contents, etc. of the state-word received and of the kind of picture related to it. On the other hand, the Picture Modifying Program involves only those decisions which modify the Picture File /or files, one at a time/ as a consequence of the information brought by the state-word received from the process.

Fig. 3.6 illustrates the general flow diagram of the Updating Program when the state-words arrive at the computer independently of whether there were changes in the state of components or not. It indicates the sequence of operations to be carried out by the computer operating system each time a state-word arises from the process, in order to update in real time the picture files related to it. It represents one of possible versions which can be implemented for the system.

The whole Updating Program includes many different operations all of them closely interrelated. The computer operating system must be able to perform all required operations at a speed compatible with the requirements imposed by the application and with a very high reliability.

The description of operations to be carried out by the Updating Program is given below.

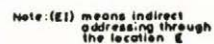


Fig. 3.6

1. Searching for the Component Vector

By using the Identifier-word arising from the process as an address for the "Pointer List", the corresponding Component Vector is searched for in the "Vector List" by indirect addressing, i.e. the Component Vector is pointed to by the contents of the location addressed in the Pointer List by the Identifier-word received.

Details about the Component Vector have been given in Sect. 3.6.2.1.2. The form in which Component Vectors are stored in the computer memory has been shown in Fig. 3.4.

2. Searching for the Picture File Address and the address of the related PMP

By using the datum "Code of Picture" stored in the previously searched Component Vector, the Picture File Address and the address of the associated Picture Modifying Program are searched for. This is carried out by means of a computer table /the Table Alfa/ using the code selected for the picture. This code is created by the Assembler starting from the "Name" given to the picture by the programmer.

3. Determination of the Type of Format

By using the datum "Type of Format" stored in the Component Vector, the type of format of the state-word is found. Depending on this datum, the Component Vector could be composed of two or more computer words.

Details about the types of format were given in Section 3.6.2.1.2.

4. Calculation of statement's address in the Picture File
and of the cell coordinate values on the useful surface

By means of an always positive "Displacement" value stored in the Component Vector, the address of the statement /in the Picture File/ and the coordinate values of the cell on the useful surface /both related to the component/ are calculated. This value represents a relative displacement to the initial address of the Virgin Program, i.e. to the Picture File Address. The contents of this initial address is the absolute coordinate values of the cell to which all the other cells composing the picture are relatively related. It is given by the "Absolute Position" Statement which compulsorily must be the first statement in every Virgin Program /See Sect. 4.4 /. The displacement of each component's statement /stored in the Component Vector/ is calculated by the Assembler based on the number and types of foregoing statements included in the Virgin Program.

Adding the value of displacement to the Picture File Address, the address of the related statement in the Picture File is calculated /Indirect Relative Addressing/. On the other hand, adding algebraically the contents of the first location of the Picture File /absolute coordinate values of the initial cell of the picture/, and the relative coordinate changes in abscissa and ordinate values /if any/ caused by each type of statement starting from the second upto the previous one to that related, the coordinate values of the cell on the useful surface can be calculated. This calculation is performed separately for abscissa and ordinate values by a special computer program.

This operation implicitly implies the standardization of all the graphical elements used to compose the drawings. Standardization has been implicitly considered when the use of graphical subroutines was proposed.

When a datum is stored in a Picture File, but does not appear visually on the screen, then the calculation of the coordinate values of the cell is superfluous and therefore is not realized. This is indicated by the datum "Type of Format" of the Component Vector and is the case of the "Z" Values stored at the end of the Picture File.

After the calculation of the statement's address in the Picture File, it is stored in a predefined register.

5. Testing of state-word bits

By using the datum "Type of Component" /or "Type of Element"/ stored in the Component Vector, the corresponding routine created to test the bits of state-words having discrete codes is called up. If a common format is given to all state-words having discrete code independently of the possible types of components, then this datum is not required.

Based on the values of bits being tested, an updated statement is formatted from the non-updated statement. In general, this formatting consists in the substitution of the previous color code by the new one and/or the complementing of the least significant bit /lsb/ of the previous symbol code, i.e. that of the non-updated statement. Whether the lsb must be complemented or not is indicated by a particular bit of the state-word. Thus, a closed switch can be easily substituted on the screen by an open switch, or viceversa, if both symbol codes are consecutive and only have a difference in the least significant bit. Other bits of state-words could be used in similar form to produce other effects.

Now, if the updated statement is the same as the non-updated statement, then the control is given again to the Updating Program and a new Identifier-word is read from the buffer. If not, the updated statement is stored in the corresponding

place in the Picture File.

If the state-word represents a numeric code, whether it belongs to a variable's value to be shown on the screen or not, similar operations are carried out. In this case, of course, the datum "Type of Component" is meaningless. The only difference is that in this case the updating value is not formatted from the previously stored value, but is processed independently. When the assembling of the Virgin Program is carried out, the Assembler leaves free locations in rigid order at the end of the file to store the information required. The corresponding displacement of these locations with respect to the first one of the picture file, is also the first time calculated by the Assembler and stored in the respective part of the corresponding Component Vector created for the component. In case of graphical subroutines, the initial address of the information related at the end of the file is stored as the third word of its data structure /See later Sect. 4.4.4/.

6. Sending the updated information to display units

Once an updated statement has been compared with the non-updated one stored in the Picture File /and stored therein substituting the previous one if they differ/, then the control is transferred to the associated Picture Modifying Program. It now modifies the remainder statements in the VP which require to be modified as a consequence of the previously updated statement. Each time a statement is conveniently modified, it is sent to the particular display unit which shows in that moment /if any/ the picture to which the statement belongs. Alternatively, the updated statements could be momentarily stored in some reserved area of the computer memory and finally sent all together to the particular display unit showing them. This apparently looks like the most advantageous solution.

When some picture is requested by a display unit, then its display address, display code and/or display number are found on a particular computer Table /the Table Beta/. This Table is updated each time a display unit calls up a new picture. If the picture has not been requested, then the control is transferred again to the Updating Program.

In order to get a high data compression in the computer memory, the statements of the Virgin Program are stored in a different form than that which would be comprehensible to the display units. Thus, every updated statement requires a code transformation before they are sent to the display units.

Updated statements are always sent to display units together with the position information of the cell related to them on the screen. These coordinate values are calculated by a special program called up by the PMP each time a VP's statement is being updated. Nevertheless, several algorithms could be created for determining the particular cell concerned each time.

The reason for which the cell coordinate values on the screen are not proposed to be stored permanently in the Picture File preceding each statement of the Virgin Program, is only that of saving computer memory locations.

CHAPTER 4

GRAPHICAL LANGUAGE

4.1 Introduction

The use of some means which permit easy communication between the operator and the industrial process being monitored is indispensable. However, conditions must be created in order to apply advantageously as far as possible the results and conclusions obtained after the study, analysis and discussion of the particular properties and requirements related with the application which we are dealing with. Among other reasons, this implied the creation of a Graphical Language, its syntax, the coding and the respective limitations and rules to be used in creating the drawings required. Thus, this Chapter complements and completes with the detailed description of the Graphical Language created, all that is necessary in order to monitor and control interactively an industrial process from the screen of a color cell-organized raster-scan graphical display. Obviously, this Chapter is consistent with all stated in Chapters 2 and 3 above. Finally, a complete example of a Pump Station has been given in order to illustrate the use of the types of statements proposed.

4.2 Necessity of a Graphical Language

Displays in control process applications are considered to have 2 major uses:

- To create and show pictures
- To operate with them.

These uses imply the display working as an input-output equipment for the digital computer. In both cases the opera-

tor of the display console constitutes an essential part of the complete system, for he follows according to his objectives the procedure established in creating pictures and when operating with the particular picture shown by the display unit, he analyzes it to take, when required, the most convenient decisions.

Creation of pictures must be a task carried out off-line in order to avoid the obstruction of the normal operation of the computer. On the other hand, operation with them must be carried out on-line, so that the process and the computer can be influenced in real time by the decisions taken each time by the operator from the display console.

In creating the pictures, the following alternatives can be used by the operator, namely:

1. Using some means which permit him to select directly the symbols of components /i.e. valves, pumps, and so on/ to be included in the picture.

In this case there must directly be available to the operator all the symbols corresponding to all possible different graphical elements to be used in creating the particular picture. This can be accomplished either by using a special keyboard with the symbols labeled over each key, or by creating on the screen a bulky "menu" with all available symbols arranged conveniently for selecting each time that required to create the picture, or creating manually their particular codes by means of some keys, each one corresponding to each bit position of the code, etc. On the one hand, due to the relatively high number of different symbols required for each picture, to the limited size of the useful surface of the display's screen, to the additional capacity required to store the "menu" into the display's refresh memory and others, the "menu" solution is here neglected.

On the other hand, by using one key for each bit position of the code /considering equal-length codes as the solution used/ and to create manually each time the code of the symbols required for the picture implies a quite time-consuming task for the operator and easily committed errors; to "prepare" the picture cell by cell giving also each time the corresponding color to the symbols; to have to know previously how those complex symbols which occupy more than one cell on the screen can be constructed with the Symbol-set available for the equipment, and so on. All these inconveniences, among others, lead us to neglect this solution also.

Therefore, from these three possibilities the use of the special keyboard with all the available symbols labelled in different keys as input means seems more commendable.

The use of this solution to create pictures would present the following advantages:

- The task can be carried out in a similar form as when an alphanumerical display is used for editing, having thus all its advantages.
- It requires a simple coloured sketch drawn on paper of the picture to be created, it being necessary only to reproduce it directly.
- It permits us to create directly any type of mentally conceivable picture without taking into account special geometric rules, rigid formats, particular instructions, etc.
- It does not require the contribution neither of the computer nor of the Controller in "editing" the pictures, since they can be prepared off-line in a stand-alone display.
- It permits the operator to perform continuously and directly the visual correction of the picture being

created as in the case of alphanumerical displays.

- After code conversion to the graphical code designed for the display itself, the transmission of the information to the computer memory can be performed in a straightforward form, but in a cell-by-cell and orderly fashion.
- It does not require highly trained operators.

Disadvantages of this solution are:

- It requires an additional labelling for the keyboard's keys commonly used for editing, or otherwise, a specially designed independent one to be used only to create the pictures.
- The operator needs to know previously how complex symbols occupying several adjacent cells can be constructed with the Symbol-set available for the equipment.
- It makes it difficult to do reverse code conversion in order to permit the use by the system of those graphical subroutines created to represent graphically the complex symbols requiring several adjacent cells, or any other type of compressed instruction.
- It does not permit the creation of new graphical subroutines such as those which can be created and used when particular parts of given pictures are repeated several times, or a new complex symbol is included in the picture, etc.
- It would require a constant memory capacity per picture to be programmed, independently of the characteristics of the picture /complexity, proportion of straight lines respect to alphanumeric characters or single symbols, etc./, that is, it does not permit us to save computer memory locations by means of a given data-compression solution.

- It requires some artistic aptitude and aesthetic appreciation from the operator /or from the programmer/ mainly when a complex symbol composed of many single symbols must be included in the picture /i.e. a pump symbol/, or when a complex drawing must be realized.
- It would require a considerable time from the operator.

We will now analyze a second alternative.

2. By using a specially created graphical language having all the features necessary for programming all drawings required by the application.

This alternative requires the creation of either an Interpreter or an Assembler with which the computer could interpret and assemble the statements written /according to the particular syntax rules and semantics of the language created/ to the understandable machine code required by the operating system and display units.

Additionally to overcoming the disadvantages of the previous alternative, the use of this alternative to create pictures presents the following advantages:

- It can be edited by using the common general-purpose keyboard used for editing in alphanumerical displays.
- By help of either a well-designed Assembler or Interpreter, it is easy to accomplish all possible code conversions either related to common graphical subroutines or related to other programmable subroutines of any type. Similarly, this can be pointed out also for compressed graphical instructions such as those for constructing horizontal and vertical straight lines.
- It permits to programmers the preparation off-line in paper of picture programs /starting also from the simple coloured sketch drawn previously/ sitting in a distant

place from the display unit, requiring only to use it when the written program needs to be stored in the refresh memory, to be corrected, to be transmitted to the computer and to debug visually the assembled picture. It permits us also to store the coded picture programs on any other type of supporting media /punched paper tape, magnetic tape, punched card, etc./ and to input them to the computer when required by using the corresponding input peripheral equipment.

- When assembled, the picture program could be stored in the computer memory in a compressed form, therefore it should permit us to save computer memory locations per picture to be stored. After debugging, if the picture program was stored in a provisional place of the computer memory /which is maybe commendable because only one program is normally assembled, debugged or modified at a time/, the transmission to its definitive place is carried out internally by the operating system when the corresponding commands have been given by the programmer from the keyboard of the specially dedicated display. Finally, the picture file prepared can easily be called up in order to be updated, permitting us also to perform easily the corresponding modifications, expansions, re-orderings, etc. Additionally, some security means can be created in the system, in order to avoid the possibility of accidental or badly intentioned modifications.
- It permits more flexibility in creating non-common drawings such as bar diagrams, records of any type, etc.
- It can be programmed in a very straightforward fashion, without need to have to know special properties about the dynamics of the part of the process symbolically represented by the picture.

Disadvantages of this second alternative are:

- It requires computer time to show graphically the

drawings to the programmer when debugging is carried out, due to the indispensable assembling required.

- It requires specially trained programmers to prepare the programs and debug them.
- It requires additional subroutines for altering /from the keyboard of the display unit specially destined for these tasks/ the statements belonging to each program stored in the computer memory. This can be accomplished by an Editor commonly used in computer systems.
- The written program can normally be interpreted only by the programmers and by the personnel intimately familiarized with the Graphical Language created.

Finally, another alternative is possible.

3. By using a language oriented to the application based on the description of the process, plant, etc.

This alternative requires the creation of a high level compiler with which the computer could prepare the statements /according to particular rules previously created/ which are further processed, optimized and finally converted to a comprehensive code for the operating system and for the display units.

The use of this alternative to create pictures presents the advantages pointed out above for the second alternative, and additionally the following ones:

- It permits a relatively easy direct interpretation of written programs by the programmers and operators.
- It could give a more logical description of the dynamics of the process and of the interaction or interrelation among the components.
- It could be topologically optimized by computing to get

aesthetics, uniform distributions, etc. among the components, following certain rules previously considered when the optimizing program is prepared.

- Maybe the picture programs will be shorter, although the highest possible compression of data can not be ensured.

Disadvantages of this third alternative are:

- It requires more computer time than the second alternative to show graphically the drawings to the programmer, because of the indispensable interpretation and further optimization of the written program.
- It requires trained programmers with considerable preparation about the specific application and with more knowledge about the dynamics of the process.
- It requires additional programs /or routines/ for altering the optimized statements from the keyboard, when debugging is carried out on those programs already interpreted by the computer and stored in the refresh memory of the dedicated display unit.
- It should use many different types of statements for specifying the great variety of possibilities encountered in actual processes.
- It requires a more elaborate Interpreter and the use of more difficult algorithms in order to interpret without ambiguities the statements composing the picture's program.

Analyzing these alternatives, due mainly to the intellectual requirements from the operator, to the larger consumption of computer time required by the computer for its interpretation and to the disputable real necessity of such an application-oriented language in this application where the drawings are normally programmed once /the first time/ and where they do

not present a great complexity in order to get aesthetics, uniformity, etc., the use of the third alternative could not be justifiable.

Comparing the advantages and disadvantages of the first and second alternatives pointed out above, clearly the second one presents more flexibility and a preferable performance to the first one. Therefore, the use in this application of a specially created graphical language should be commendable. Of course, this implies the creation of the minimal number of statements to be used by the programmers for preparing the programs /that is, the language itself, its syntax, etc./ and the creation of the Assembler which handles them and performs the necessary code conversions.

The flow of the information when a specially dedicated display unit or any other type of input-output peripheral equipment is used to create pictures by means of the statements of a graphical language, has been shown schematically in Fig. 3.3. /See Sect. 3.6.1.1./

4.3 Creation of Pictures

Taking into account the common topological characteristics numbers 1, 5, 6, 7, 10, 12, 13 and 15 of drawings enumerated in Sect. 2.5, we suppose that creating a unique drawing to represent each independent part of the complete plant, it will be possible to supervise and eventually to control in a relatively easy form, the components included inside the corresponding section, area, etc. of the plant.

Characteristic 1 permits us to prepare a sketch of the drawing on a non-special sheet of paper with particular physical dimensions and cell subdivisions.

Characteristics 5, 6 and 7 permit the distribution and

allocation of the component symbols in the most convenient form.

Characteristic 10 permits the use of cell subdivision of the complete drawing, each cell having only one content at a time for each different drawing /similarly as in alphanumerical displays./

Characteristics 12 and 13 permit us to build-up independent drawings for each area, zone, section, type of network, etc. of the plant.

Characteristic 15 always permits them to be drawn initially with only one color and on a single-color background.

With all these in mind, we can affirm that all drawings could be prepared:

- off-line,
- only once for each part, section, etc. of a given industrial process,
- with a definite and fixed maximal number of cells depending on the particular cell's dimensions and on the total screen surface of the display,
- without special physical means,
- using a limited number of predefined graphical symbols included into a specially designed Symbol-set,
- knowing some relatively simple particular dynamic characteristics of the section, area, etc. being symbolically represented by the drawing,
- by personnel without a very high level of preparation.

4.3.1 Categories of the visual information to be handled by the display

In designing and implementing the most appropriate graphical language to be used in this application, requirements must be analyzed in order to get a maximal efficiency when programs of drawings are prepared and to provide to programmers a powerful tool which has flexibility and permits them to carry out the programming task easily.

In order to get the major advantages of a cell-organized raster display and because of the particular characteristics of drawings required in process control applications /See Sect. 2.5/, the use as far as possible of the repetition facility shall be a main goal.

First of all, the analysis of the types and characteristics of the graphical information to be included in pictures and handled by the display system, permits us to group it in four well-differenciated categories, namely:

- invariable graphical information /explicitly shown/ always with the same color /i.e. legends, designation of drawings, single-state passive components, etc./;
- color-variable information /explicitly shown/.
This case take place when the condition of a component varies, but the symbol remains invariable /i.e. closed or opened valve, pipelines conducting fluids or not, etc./;
- information variable in nature /explicitly shown/.
This case take place when the state of a component varies, but the color remains invariable /i.e. closed and opened switch, time information, written alarm information, indications of instruments, etc./;
- information variable in color and variable in nature /explicitly shown/.



This case takes place when to given symbols of different nature there are associated predefined color codes to indicate abnormal situations /i.e. switch remaining open after a particular command has been given to close it, out-of-range indications of measuring instruments, and so on/.

Taking into account all these factors, a Graphical Language specially created for the application is proposed. First of all, some requirements are established.

4.3.2 Requirements for the Graphical Language

Taking into account the factors stated above and according to the general conception for the display system proposed in the present study /See Sect. 3.6.1.1/, the requirements which have been considered as sufficient for the Graphical Language to be used in this application are:

1. It must permit the definition of the absolute coordinate values /abscissa and ordinate/ of any cell located within the useful surface.
2. It must permit the handling of coordinate values, either once or repetitively, referred to any statement used to define graphical information. At the same time, it must permit us to realize "movements" from any arbitrary cell to any other cell within the useful surface.
3. It must permit us to select easily the color of any graphical information, either explicitly or as a product of computer decisions.
4. It must permit the inclusion of data supplied by the programmer, coded information obtained either from the processing of the information received from the plant in state-words or possibly given by operator commands, etc., and create the conditions for an easy updating.

5. It must permit the repetition of any graphical information by means of a single statement at least up to 16 times /See Sect. 2.6.1/.
6. It must permit the preparation of explicit texts of any length, with any color and using a minimal number of instructions.
7. It must permit easily the composition of complex symbols from those single symbols included in the Symbol-set and handle them as single elements /See Sect. 2.5.2/.
8. It must permit us to refer the whole picture relatively to a particular absolute position explicitly given by the programmer.
9. It must permit the creation of all types of drawings required by the application and their updating when required.
10. It must permit us to develop any graphical information horizontally as well as vertically, and prepare the conditions of the system to locate the next graphical information beginning in the following cell adjacent to the particular cell where the previous one ended.
11. It must permit a high data compression of the information to be handled in order to save memory locations in the operative memory of the computer, that is, it must have an appropriate data structure.
12. It must include the minimal possible number of statements of different type with a logical syntax and use an easy semantics which facilitates the use of them by the programmer.
13. It must indicate the termination of the Virgin Program and of the Picture File wherein this is stored.
14. It must indicate the termination of the called-up graphical subroutines in order to return the control to the Virgin Program.

15. It must be flexible in its use and easy to learn.
16. Statements to be used must be simple and easy to interpret.
17. It should allow on-line interaction between the operator and the diverse parts of the whole system.

Requirements enumerated above impose also in some measure the requirements for the Interpreter which handles the statements of the Graphical Language created.

4.4 Description of the Graphical Language

4.4.1 Conventions

In writing those programs which produce the desired pictures on the screen of the display units by means of a Graphical Language, it is advantageous to use all the facilities considered previously as necessary for the whole system in order to create in the simplest form the drawings required by the application.

First of all, two conventions are stated in connection with the numbering of the cells of an arbitrary rectangular useful surface "S" /Fig. 4.1/, namely:

1. The cell having Cartesian rectangular coordinates (0,0) is considered as the first left-hand side cell along the bottom edge. In this form, we are consistent with the normal convention.
2. The cells are consecutively numbered from left to right in all the rows and from bottom to top in all the columns.

As a result of these conventions we have:

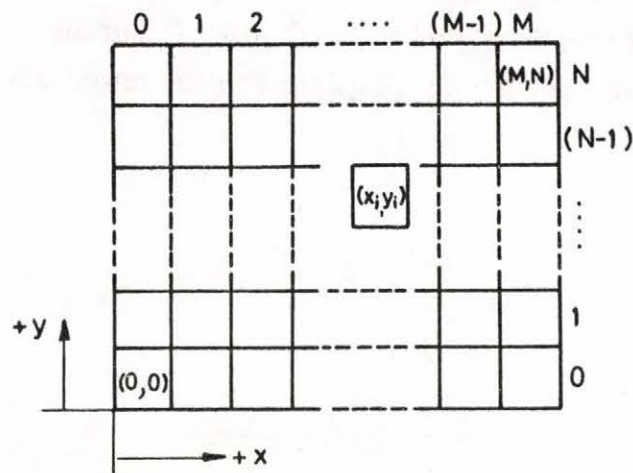


Fig.4.1

1. All the absolute rectangular coordinates of cells will always have positive values. Therefore, if there are "M" cells per row and "N" cells per column, then the following inequalities are always fulfilled.

$$0 \leq x_i \leq M$$

$$0 \leq y_i \leq N$$

2. When the last raster takes place in a row of cells and it arrives to the last cell in the row, /cells with coordinates M, y_i /, the position information of the next cell /indicated by " x'_i " and " y'_i " / is given by:

$$y'_i = y_i - 1$$

$$x'_i = 0$$

In case of $x_i = M$ and $y_i = 0$, then the coordinate values of the next cell are given by:

$$y_i = N$$

$$x_1 = 0$$

3. Changes in coordinate values of an arbitrary cell having absolute /or relative/ coordinates (x_1, y_1) are performed according to Fig. 4.2 for each direction and sense indicated.

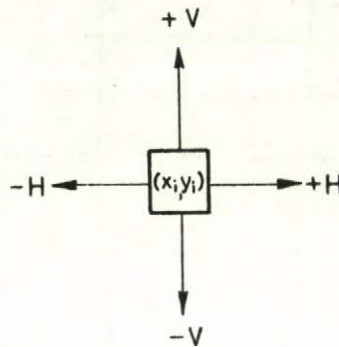


Fig. 4.2

In Fig. 4.2:

- $+H$ means to add unity to the abscissa value of the current relative coordinate " x_1 "
- $-H$ means to subtract unity from the abscissa value
- $+V$ means to add unity to the ordinate value of the current relative coordinate " y_1 "
- $-V$ means to subtract unity from the ordinate value.

4.4.2 Types of Statements

The following types of Statements are proposed for the Graphical Language in order to fulfil the requirements pointed out in Sect. 4.3.2:

1. "Absolute Position" Statement

2. Graphical Statements

2.1 "Text" Statement

2.2 "Element" Statement

2.3 "Call to Subroutine" Statement

3. "Move" Statements

3.1 "Move without Repetition" Statement

3.2 "Move with Repetition" Statement

4. "Declare Subroutine" Statement

5. "End of Subroutine" Statement

6. "End of Program" Statement

7. "End of File" Statement

As a rule, only one statement per row can be specified in programs created with the Graphical Language.

Now, the effect and restrictions of each type of statement together with the corresponding syntactic format is given below.

4.4.2.1 "Absolute Position" Statement

This type of statement is used to indicate absolutely the rectangular coordinates of the initial cell to which all the succeeding statements are relatively referred to. In general, the position is considered to be of two different classes, absolute position and relative position. The absolute position always has its own meaning, that is, "Absolute Position" statements are always self-defined; on the other hand, the relative position does not have a meaning of its own, for it is always associated to the actual current coordinate values which in their turn, are referred to the absolute position information explicitly indicated in the program.

The general syntactic format of the "Absolute Position"

statement is:

PX,Y

where:

P indicates "Absolute Position" statement

X,Y indicate the absolute abscissa "x" and ordinate "y" values of the cell to which the subsequent statements are referred. These values are always positive values.

The comma between abscissa and ordinate is used only to separate one from the other. A "space" character is compulsorily required after the identifying letter "P". The "CR" character /Carriage Return/ indicates the end of the statement to the Interpreter.

Every picture or drawing must have as first statement of the program which creates it, an "Absolute Position" statement to which all subsequent statements will further be relatively referred. Therefore, if one statement of this type is not written as the first one in the program, an error is indicated to the programmer. In this case the program does not run.

Additionally, an "Absolute Position" statement must be compulsorily followed by a Graphical Statement. In any other case, there is a syntactic error which must be indicated in some form by the Interpreter.

This type of statement fulfills the first requirement of the Graphical Language.

4.4.2.2 "Graphical" Statements. Generalities

These types of statement are used to define the graphical

information to be located in one or more cells starting from the cell specified either by the coordinates of the previous "Absolute Position" statement or by the current implicit relative position information.

Each Graphical statement defines unambiguously only one graphical element. A graphical element is anything which can be created with the characters and single symbols included in the Character- and Symbol-sets of the system, using only one graphical statement.

Graphical statements are completely specified by means of two components, namely:

- the parameters
- the contents

The parameters will be analyzed first.

Parameters. To every contents are always associated a group of parameters. They refer to information about:

1. Orientation
2. Color
3. Repetition

1. Orientation

Orientation is specified by both the direction and the sense. The direction can be horizontal and vertical; the sense is specified for each direction by means of the plus and minus arithmetical signs. Thus, four cases are possible:

- +H horizontal to the right
- H horizontal to the left
- +V vertical to the top
- V vertical to the bottom

These directions and senses are shown graphically in Fig. 4.2 and their meanings and actions correspond with conventions adopted before /See Sect. 4.4.1/.

The orientation parameter refers to the direction and sense where the next cell to be positioned is located with respect to the current cell, and does not refer to the orientation of cells which is always the same. It intrinsically indicates the coordinate value /abscissa or ordinate/ which must be incremented or decremented of either the previously specified absolute or actual current relative position information.

In statements which use it, the orientation is always explicitly indicated by the signs "+" and "-" followed by the letters "H" or "V", thus:

- +H indicates that the next cell to be positioned each time is located in the same row and to the right of the current cell
- H identical to the left
- +V indicates that the next cell to be positioned each time is located in the same column and above the current cell
- V identical below

They are encoded by using only two information bits on each type of graphical statement.

This parameter is the first one indicated within the group of three possible parameters. In its most general form it is noted in the syntax of statements as "sd", where the letter "s" means sense /or sign/ and the letter "d" means direction.

2. Color

Color facilities are eight according to the color CRT types proposed /Shadow-mask or Trinitron Tubes/ [74], namely: blue, green, yellow, red, cyan, magenta, white and black. Each color will have a given meaning according to conventions previously stated for the system. Black color takes place when the electron beam is blanked. This facility is used when a particular cell /or cells/ are destined to receive some particular information previously considered /or maybe not/ in the picture program, e.g. when some component falls in abnormal situation for showing to operators the legend OUT OF SERVICE, etc.

The color information represents a quite important parameter generally resulting from the analysis by program of the state-words received from the industrial plant. It could have different meanings for the operator depending on the conventions previously stated in the different types of drawings considered. It is encoded by using only three information bits in each type of graphical statement.

In programming, the color is always explicitly indicated by means of particular alphabet letters. These are the following:

B blue	C cyan
G green	M magenta
Y yellow	W white
R red	N black

This parameter is the second one indicated within the group of three possible parameters. The color is noted in the syntax of statements by the letter "c" /color/.

3. Repetition

This parameter indicates the integer number of times it is desired to repeat the contents of a graphical statement. The maximal integer number of repetitions permitted for the graphical elements is 16 times, thus, to indicate it only 4 information bits per graphical statement are required.

This facility in the system is particularly useful in creating any type of straight lines, for shading of areas, in creating special complex symbols, repeated texts, and so on.

This parameter is indicated in statements by means of a decimal number in the range $0 \leq r \leq 15$, since it has only 16 different possibilities.

This parameter occupies the last position within the group of three parameters. The repetition is noted in the syntax of statements by the letter "r" /repetition/.

The group of parameters is always indicated in graphical statements rigidly ordered, enclosed in circular brackets, and separated one from each other by commas. They have effect or validity over all the information specified as the contents in the graphical statement to which the parameters belong.

The group of parameters, when included in graphical statements, permits us to fulfil the requirements numbers 3, 5, and 10 of the Graphical Language.

Contents. In graphical statements the contents is specified following the parameters, but separated by one "space" character. It consists in the explicit specification of the

"Name" of the graphical element for the identification of the graphical information to be shown.

The contents permit us to specify unambiguously the graphical elements required for composing the drawings. Together with the parameters, they constitute all that is necessary for specifying in programs any type of graphical information required by the application.

Combination of these data /contents and parameters/ makes possible the definition of a practically infinite number of different graphical elements with a finite number of characters and single symbols.

Graphical statements are of three different types:

- "Text" statement
- "Element" statement
- "Call to Subroutine" statement

Each type of statement has its own properties.

4.4.2.2.1 "Text" Statement

Texts refer to that graphical information which must be totally interpreted and depicted in the same form it is written or specified in the graphical statement. It does not require any transformation, previous calculations, etc., that is, it consists in the information which can be directly written by means of the actual meaning or labelling of the Keyboard's keys.

Text information can only be horizontally or vertically developed on the display's screen using only one statement. The position information of each subsequent character, punctuation mark, etc. within a "Text" statement is calculated each

time from the relative position information of the preceding one, after it has been conveniently modified. The first character of "Text" statements can be positioned either absolutely or relatively.

In writing texts in different rows or columns, the corresponding information must be ordered as it must appear in each row or column.

"Text" statements do not necessarily need to be preceded by any other particular type of statements if sometime before an "Absolute Position" statement has been formally given.

The general syntactic format of the "Text" statement is:

$$T(sd, c, r) \text{ TEXT}$$

where:

T	indicates "Text" statement
(sd, c, r)	indicates the group of parameters for the text
sd:	orientation
c:	color
r:	repetition
TEXT	indicates the text itself.

A "space" character is compulsorily required after the identifying letter "T". The "CR" character /Carriage Return/ indicates the end of the statement.

Thus, "Text" statements can be used in writing texts, for tabulation in Tables, lists, etc., to repeat single alphanumeric characters and editing symbols, to reserve free spaces or zones, for deleting, and so on. All these can be specified either vertically or horizontally.

As a result of the real needs, the repetition parameter in "Text" statements is permitted to acquire only values in the range

$$0 \leq r \leq 7$$

Thus, an additional information bit is available for other uses.

The "Text" statement fulfills the requirements numbers 4 and 5 for the Graphical Language.

4.4.2.2.2 "Element" Statement

The "Element" term consists in graphical information which can be unambiguously recognized by a "Name". The "Name" in this type of statement constitutes the contents of the "Element" statement, which normally identifies a particular symbol in the ROM /or PROM/ included in the hardware of the display unit wherein the Character- and Symbol-sets are stored. They comprise those statements which graphically create a single element occupying one or more /upto 16/ consecutive cells either horizontally or vertically. The "Element" statement does not necessarily need to be preceded by any other particular type of statement, if sometime before an "Absolute Position" statement has been given.

The general syntactic format of the "Element" statement is:

$E \sqcup (sd, c, r) \sqcup \text{NAME}$

where:

E indicates "Element" statement
(sd, c, r) indicates the group of parameters for the graphical element whose "Name" is specified in the statement
sd: orientation; c: color; r: repetition

n indicates the type of cutting of the symbol. It is always specified by a decimal number /See Sect. 2.5.3.4 and Appendix 1/. If the graphical element is required as a whole, i.e. without cutting, then this datum must be omitted when the statement is specified.

NAME indicates the "Name" of the graphical element.

A "space" character is compulsorily required after the identifying letter "E" and between the modifier and the "Name" of the graphical element. The "CR" character /Carriage Return/ indicates the end of the statement.

The "Name" of the graphical element can be a sequence of 12 characters as maximum /either letters or numerals, but the first one must be a letter/ by means of which the different elements can be unambiguously identified. In this application, the characteristics of components composing the drawings permit us to select self-explanatory "Names" for the different symbols required. If erroneously a non-existent "Name" is used in an "Element" statement, i.e. it does not identify one of the single graphical elements available, then the Interpreter must signalize an error.

The "Element" statement contributes to fulfil the requirements 7, 9 and 11 for the Graphical Language.

4.4.2.2.3 "Call to Subroutine" Statement

This type of statement permits calling up the graphical subroutines created for the system.

In the present study, a graphical subroutine is a group of

statements of the Graphical Language by means of which a complex symbol can be created, i.e. a symbol which occupies two or more single cells on the useful surface. Conditions must be created to define an unlimited arbitrary number of graphical subroutines available for all the drawings of the system. Every subroutine is identified by a "Name".

Previously to calling a graphical subroutine, the initial address must be declared and stored in some location directly or indirectly accessible to all the drawings. This declaration is carried out by means of a "Declare Subroutine" statement /See Sect. 4.4.2.4/. Every graphical subroutine must be ended by means of an "End of Subroutine" Statement /See Sect. 4.4.2.5/. Statements composing a graphical subroutine can be of any type, except of the "Absolute Position" type, i.e. this type of statement may never be specified among the statements composing a graphical subroutine.

A "Call to Subroutine" Statement need not necessarily be preceded by any other particular type of statement if sometime before an "Absolute Position" statement has been formally given, and provided that it has been correctly declared previously by means of a "Declare Subroutine" statement.

Graphical subroutines do not require cutting, orientation or repetition parameters when they are specified, but only the color parameter. On the other hand, they require by nature either transformations, breaking up, calculations, or other kinds of processing.

The general syntactic format of the "Call to Subroutine" statement is:

CL(c)NAME

where:

- C indicates "Call to Subroutine" statement
- c indicates color parameter. It must always appear enclosed within circular brackets
- NAME indicates the "Name" of the subroutine.

A "space" character is compulsorily required after the identifying letter "C". The "CR" character /Carriage Return/ indicates the end of the statement to the Interpreter.

The "Name" of a graphical subroutine can be a sequence of a maximum of 12 characters /either letters or numerals, but the first one must be a letter/, by means of which it can be identified among the remainder. If erroneously the same "Name" has been given to two different subroutines or a graphical subroutine has been declared two or more times, then the Interpreter must signalize an error. Similarly, an error is signalized when a non-existent "Name" is specified in a "Call to Subroutine" statement.

"Call to Subroutine" statements specified with black color "N" must be interpreted but not transmitted to display units. In this form we may avoid the representation on the useful surface of those graphical statements of the graphical subroutine which have explicitly specified the color /and not the designation [C]- See Sect. 4.4.3.1.2 - by means of which it acquires the color specified in the "Call to Subroutine" statement/.

The "Call to Subroutine" statement contributes to fulfil the requirements numbers 7, 9 and 11 for the Graphical Language.

4.4.2.3 "Move Statement"

This type of statement is used to repeat the graphical statement preceding it changing each time the coordinate values

of the current relative position information. In general, it only permits us to perform arithmetic operations /addition and subtraction/, on the current abscissa and ordinate values of any relative coordinates referred to some absolute position information formerly given.

It is particularly useful also in creating complex symbols, special graphical elements, in repeating any type of graphical element a given number of times without altering other parameters, etc.

The general syntactic format of the "Move" statements is:

$$MUR(sX,sY)$$

where:

M indicates a "Move" statement

R indicates the integer number of times that the signed coordinate values enclosed in circular brackets following it, must be repeatedly operated /i.e. added or subtracted/ each time on the last current relative position coordinates, before the precedent graphical statement is considered again

(sX,sY) indicates the signed numerical values to be correspondingly added or subtracted the number of times indicated by "R", to the current relative position information referred to some previously given absolute position information. After this, the precedent graphical statement is considered again. The commas between both signed coordinates and the circular brackets surrounding them are always compulsorily required.

A "space" character is compulsorily required after the

identifying letter "M". The "CR" character /Carriage Return/ indicates the end of the statement to the Interpreter.

"R" is merely a repetition parameter. It can acquire integer values in the range, for example, $0 \leq R \leq 63$, when the maximal number of positionable cells either in the horizontal or vertical direction is 64 cells. Thus, movements along the whole useful surface can be carried out by using only one "Move" statement.

Repetition explicitly specified by the digit zero /or simply not specified/, means that the arithmetic operations with the signed coordinates enclosed within circular brackets in the statement are not repeated at all, but are only performed once in order to alter the coordinate values of the current relative position information. In this case, a single movement on the useful surface is performed.

Repetitions with other positive integer values than zero mean that the coordinates enclosed within the circular brackets are algebraically added "R" times to the coordinate values of the current relative position information existing when the statement is given. In this case, it organizes something like a program loop.

In the first case /when $R=0$ / the graphical program continues, in the second case /when $R \neq 0$ and positive/ the previous statement is repeated "R" times each one with the new coordinate values algebraically calculated.

This type of statement will permit very easily the creation of differently colored Bar Diagrams, particular shadings etc., by using either data explicitly supplied by the operator, data called up by the computer from its operative memory, adequately coded data received from the process in state-words, etc.

The correct use of the "Move" statement takes place when it is preceded by any type of Graphical Statement either to perform movements or to repeat it a given number of times. "End of Subroutine" statements /See Sect. 4.4.2.5/ can not precede a "Move" statement.

Two consecutive "Move" statements carry us to the following alternatives:

- "Move without Repetition" statement followed by another "Move without Repetition" statement, i.e.:

M (sX,sY)

M (sX₁,Y₁)

This case is valid, but is a bad use of "Move without Repetition" statements.

- .. "Move without Repetition" statement followed by a "Move with Repetition" statement, i.e.:

M (sX,sY)

M R(sX₁,sY₁)

It is not permitted, because the "Move" statement is not a graphical statement. This case is trivial.

- "Move with Repetition" statement followed by a "Move without Repetition" statement, i.e.:

M R(sX,sY)

M (sX₁,sY₁)

This case is valid. Because the sequence of the action of "Move with Repetition" statements is first to effectuate the algebraic calculation on the coordinate values and after this to repeat the preceding graphical statement

upto "R" repetitions have been performed, then after this a movement on the useful surface could possibly be required.

- "Move with Repetition" statement followed by a "Move with Repetition" statement, i.e.:

M R(sX,sY)

M R₁(sX₁,sY₁)

This is also a trivial case for the same reason given above.

"Move" statement fulfills the requirements numbers 2 and 8 for the Graphical Language.

4.4.2.4 "Declare Subroutine" Statement

This type of statement is compulsorily required to declare each graphical subroutine to be created.

The general syntactic format of the "Declare Subroutine" statement is:

SNAME

where: S indicates "Declare Subroutine" statement
NAME indicates the "Name" given to the graphical subroutine declared.

Restrictions with respect to the "Name" of graphical subroutines are the same as those established for the "Call of Subroutine" statement's Names.

A "space" character is compulsorily required after the identifying letter "S". The "CR" character /Carriage Return/

indicates the end of the statement to the Interpreter.

The "Declare Subroutine" statement fulfills the requirement number 7 for the Graphical Language.

4.4.2.5 "End of Subroutine" Statement

This type of statement is compulsorily required to indicate the end of each graphical subroutine created.

The general syntactic format of the "End of Subroutine" statement is:

EOS

which means End of Subroutine.

Obviously, in every picture program the number of "End of Subroutine" statements is the same as the number of "Declare Subroutine" statements.

This type of statement fulfills the requirement number 14 for the Graphical Language.

4.4.2.6 "End of Program" Statement

This type of statement is compulsorily required to indicate the end of each picture program created.

The general syntactic format of the "End of Program" statement is:

EOP

which means End of Program.

Obviously, only one "End of Program" statement is required

per picture program.

This type of statement contributes to fulfil the requirement number 13 for the Graphical Language.

4.4.2.7 "End of File" Statement

This type of statement is compulsorily required to indicate the end of each Picture File opened in the system.

The general syntactic format of the "End of File" statement is:

EOF

which means End of File.

Obviously, only one "End of File" statement is required per Picture File. After it, a new Picture File can be opened for the system.

This type of statement contributes to fulfil the requirement number 13 for the Graphical Language.

4.4.3 Auxiliary Coded Information

4.4.3.1 Data, Local Variables and "Z" Values

4.4.3.1.1 Data

Data are considered in the Graphical Language as those sequence of characters which will always be explicitly given by the programmer or by the operator. They can be interpreted either as the actual numeric value of a given sequence of numerals, or simply as alphanumeric characters each with their own meaning /i.e. as numerals, letters, editing symbols, or punctuation marks/. Maximal and minimal numeric

values which can be accepted by the computer and the code in which they can be expressed /i.e. decimal, hexadecimal, or other/, depends on the length of the computer word and on the software created for the system. Eight different types are possible.

Explicit data are stored in arbitrary order within a group at the end of the Picture File between the "End of Program" and "End of File" statements. They are specified in statements by means of the letter "D" followed by a decimal number, both enclosed in square brackets, e.g. [D4], [D62], etc. The maximal number of data which can be used in the system per Picture File depends on the distribution of bits realized in the coded computer word which identifies them.

As a rule, data once have been given can not be altered. They permit us, for example, to use a graphical subroutine in different Virgin Programs having in each one of them different legends, physical units /as in case of coordinate axes/, etc. In programs, data are successively specified in rigid order /after the corresponding "Call to Subroutine" statement/ preceded by the letter D and a number indicating the type of data in this form: Dn DATUM. These data do not include those required by the Picture Modifying Programs.

The possibility of specifying explicit data contributes to fulfil the requirement number 4 for the Graphical Language.

4.4.3.1.2 Local Variables

Local Variables are considered in the Graphical Language as those values obtained as a result of calculations carried out by the Picture Modifying Program. They are referred always to a particular type of information specified in the statements of the Virgin Program related, i.e. color, orientation, etc.

When programs are prepared, in some cases as in the case of text or color specifications, the use of meaningless values

for the parameters handled by the Local Variables is of great usefulness. Types of parameters together with their corresponding meaningless values are detailed in Table 4.4.1.

Table 4.4.1

Types of parameters and corresponding meaningless values

Type of parameter	Meaningless value	Observations
arbitrary signed number	0	number zero
color	N	black color /blanked beam/
repetition "r"	0	zero repetition
text /or code/	"SP"	"space" character
relative coordinate value	0	the precedent remains invariable
Repetition "R"	0	previous Graphical statement is not repeated

Local Variables are used mainly in graphical subroutines. They permit us to use a single graphical subroutine several times in the same Virgin Program each time using different parameters, either given by the programmer as explicit data, or calculated by the Picture Modifying Program related to it, e.g. in case of a bar diagram to construct the bars with the height depending on some data received from the process, or to show on the screen in some place near to the corresponding symbol the flux rate of an operating pumping equipment, etc.

As a rule, local variables acquire different values each time the process sends the computer system a different value

of a given "Z Value" /See below in Sect. 4.4.3.1.3/.

Local Variables are stored in arbitrary order within a group at the end of the Picture File between the "End of Program" and "End of File" statements. They are specified in statements by means of the letter "Y" followed by a decimal number, both enclosed in square brackets, e.g. [Y16], [Y41], etc. As in the case of explicit data, the maximal number of local variables which can be used in the system per Picture File depends on the distribution of bits realized in the coded computer word which identifies them.

Types of parameters with the number of bits required to specify each of them are given in Table 4.4.2.

Table 4.4.2

No.	Type of parameter	Bits required
1.	orientation "sd"	2
2.	color "c"	3
3.	repetition "r"	4
4.	" ΔY " coordinate	6
5.	" ΔX " coordinate	6
6.	Repetition "R"	6
7.	Code "C"	7
8.	signed number "n"	16

In some statements composing a graphical subroutine, the color parameter can be given as a letter "C" enclosed in square brackets, i.e. as [C]. This represents a special case of a local variable. It indicates to the Interpreter that the color for the particular statement must be the color ex-

explicitly indicated in the "Call to Subroutine" statement used to call the graphical subroutine to which it belongs. This possibility is particularly useful to change the color of complex symbols according to actual conditions of the component in the plant.

The use of local variables contributes to fulfil the requirement number 4 for the Graphical Language.

4.4.3.1.3 "Z" Values

"Z" Values are considered as those values obtained as a result of calculations carried out by the System's Supervisor Program with the information received in state-words from the process. Every Picture File has its own "Z" Values. They are used by the Picture Modifying Program associated to the respective Picture File for calculating the local variables related.

"Z" Values are stored in arbitrary order at the end of the Picture File and between the "End of Program" and "End of File" statements. They are never specified in the statement of Virgin Programs. As a rule, "Z" Values are always numerical values.

The use of "Z" Values contributes to fulfil requirement number 4 for the Graphical Language.

4.4.3.2 "Mute" code

A "Mute" code is always created automatically by the Interpreter in those cases when it is required to include within the Virgin Program a computer word without any information, i.e. a computer word in the data structure which does not require to be transformed, broken-up, etc.

A particular use of the "Mute" code takes place when state-

ments of a graphical subroutine have neither local variables nor data specified. In this case, the "Mute" code is automatically stored in the place of the third word of a "Call to Subroutine" statement which specifies the address of the group of data and local variables related to /and therefore required by/ the graphical subroutine /See Fig. 4.4/.

4.4.4 Coding. Distribution of bits of statements

The Coding and distribution of bits proposed for the manipulating system is shown in Fig. 4.3 for the first computer word of each statement, and in Fig. 4.4 for successive computer words /if any/, for a useful surface with 64x48 cells.

The distribution of bits is only one of many possible alternatives, and not considered as the optimal one. The two types of instructions required by the display units are shown in Fig. 4.5.

The main reasons which gave place to the distribution of bits given are:

- to make possible in all cases the division of the whole computer word in two independent bytes having wholly particular data each,
- to locate data of the same nature in the same relative place in every computer word,
- the great number of information bits required by the "Element" statement to specify the parameters and modifiers required for it /13 from 16 bits/,
- with all the above factors in mind, in order to achieve as much modularity as possible.

Order Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 bit	Type of statement	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	"Mute" code	
2	1	[D number]						0	Type					0	0	0	Code for "Data" (64 data)	
3	0							0							0	1	0	Not used
4	1	[Z number]						0	Type					0	1	0	Code for "Z" Values (64 "Z" Values)	
5	0			color			0							1	0	0	"Call to Subroutine" statement	
6	1	[Y number]						0	Parameter					1	0	0	Code for "Local Variables" (64 L.V.)	
7	0							0							1	1	0	Not used
8	1							0							1	1	0	Not used
9	repetition		color				0	modifier				0	sd			1	"Element" statement	
10	0	rep.	color				0					0	1	sd			1	"Text" statement
11	1							0					0	1			1	Not used
12	0							0					1	1			1	Not used
13	1							0					1	1			1	Not used
14	0							1					0			0		"End of Subroutine" statement
15	1							1					0			0		"End of Program" statement
16	0							1					1			0		"End of File" statement
17	1							1					1			0		Not used
18	0	signed X						1	signed Y						0	1		"Move without Repetition" statement
19	1							1							0	1		Not used
20	0	signed X						1	signed Y						1	1		"Move with Repetition" statement
21	1							1							1	1		Not used
22	0	X							0	Y								"Absolute Position" statement

Fig. 4.3

Number of Statement	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 bit	Information related
2	Numerical value	Datum
4	Numerical value	"Z" Value
5	Address Address	Address of the graphical subroutine Initial address of group of data and local variables related
6	Parameter (high byte) ; signed number (whole word)	Local Variable's value
9	Code	Code of symbol
10	0 character 0 character 0 character 0 character ⋮ 0 character 1 last character	Characters of the text
20	Repetition	Value of Repetition

Fig. 44.

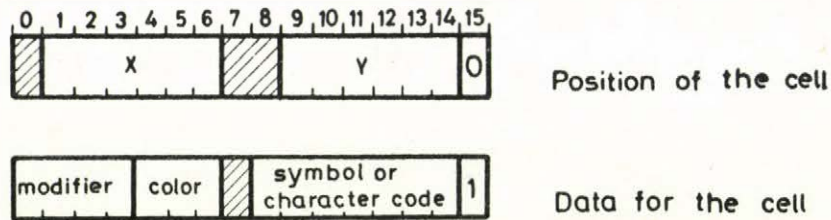


Fig. 4.5

Some characteristics of statements proposed are given in Table 4.4.3 in connection with:

1. Total number of words required by each one /1 word:
16 bits/
2. Total number of non-used information bits and percent of total
3. Minimal and maximal number of cells handled by each one
4. Limitations
5. Observations

4.5 Complete example for a Pump Station

Now, a complete example is given to illustrate the use and power of all types of statements defined above. The fulfilment of requirements numbers 9 and 12 for the Graphical Language are made evident in some measure in the formulation of the present example.

The example deals with the design of a picture for an arbitrary pump station. The color sketch of this picture /already updated/ is shown in Fig. 4.6. It consists of a tank to supply water to another part of the system not shown explicitly in the drawing. This task is carried out with help

Table 4.4.3

No.	Statement	Words required	Non-used bits Number	bits %	Cells handled		Limitation	Observations
					Min.	Max.		
1	Absolute Position	1	2	12.5	1	1	$0 \leq X \leq M$ $0 \leq Y \leq N$ Can be used only once	X: 64 cells Y: 48 cells First statement always
2	Element	2	9	28.1	1	16	-	Uses the modifiers
3	Text	$1 + \frac{c}{2}$	3	18.8	1	rc	$rc \leq X$ /if horizontal/ $rc \leq Y$ /if vertical/	r: repetition c: number of characters X: 64 cells Y: 48 cells $1 \leq r \leq 8$
4	Call to Subroutine	3	8	16.6	2	S	Has no repeti- tion orienta- tion para- meters	S: area of use- ful surface
5	Move with- out repe- tition	1	0	0	1	S	See Sect. 4.4.2.3	Handles signed numbers
6	Move with repetition	2	10	31.2	0	S	See Sect. 4.4.2.3	S: area of use- ful surface Handles signed numbers

Table 4.4.3 /Continuation/

No.	Statement	Words required	Non-used bits		Cells handled		Limitation	Observations
			Number	%	Min.	Max.		
7	End of Subroutine	1	12	75	-	-	Cannot be used within the Virgin Program itself, but at the end of each graphical sub- routine	-
8	End of Program	1	12	75	-	-	Can be used on- ly once at the end of the Virgin Program	Last state- ment of the Virgin Pro- gram
9	End of File	1	12	75	-	-	Can be used on- ly once, at the end of the Pic- ture File	Last state- ment of the Picture File
10	Declare Subroutine	0	-	-	-	-	Cannot be used within the Virgin Program itself, but at the beginning of each graphical sub- routine	-

of one of the two pumping sets shown. The conduction of water through pipe lines is controlled by valves with only two stable states: open or closed. The tank can be normally filled through a valve from another network not shown in the drawing. The tank can supply water directly to the pumping sets independently of the actual state of the supplying network. Conduction of water to the supplying network is not permitted. The water can be recirculated from the tank through the selected pumping set and again to the tank in that sense; the contrary sense is forbidden.

With all this stated, which constitutes the dynamics of the drawing, the corresponding Virgin Program and Picture Modifying Program have been prepared. The list of statements of the Virgin Program prepared is shown in Appendix 1, the list of the updated Virgin Program with the corresponding changes in the data structure is shown in Appendix 2, the non-updated display picture is shown in Fig. 4.7 /with alarm legends included/, the updated display picture is shown in Fig. 4.8 and one solution of many possible for the general flow diagram of the Picture Modifying Program related is shown in Fig. 4.9. In this, change in color of valve and pump symbols is carried out directly by the General Updating Program /See Sect. 3.6.3.4 and Fig. 3.6/, based on the information received in state-words from the industrial process. Thus, these changes are not shown in Fig. 4.9. The information flow followed in updating the updated drawing of Fig. 4.6 is shown with thick lines in Fig. 4.9.

The color used for the symbols of the Virgin Program and for the variable information in general in this example /except for the time/ have been the red color, indicated in statements by means of the letter "R" in the particular place of the color parameter. For the invariable /or fixed/ information we have used the blue color indicated with the letter "B".

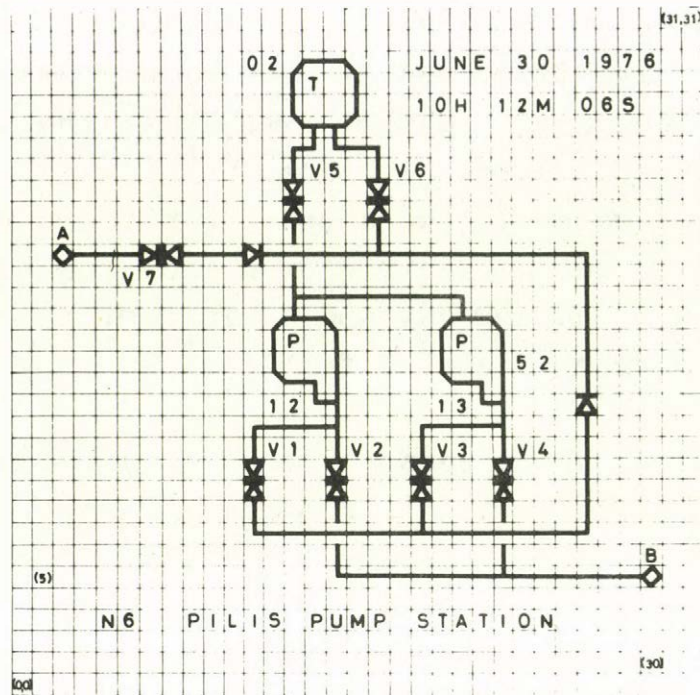


Fig. 4.6

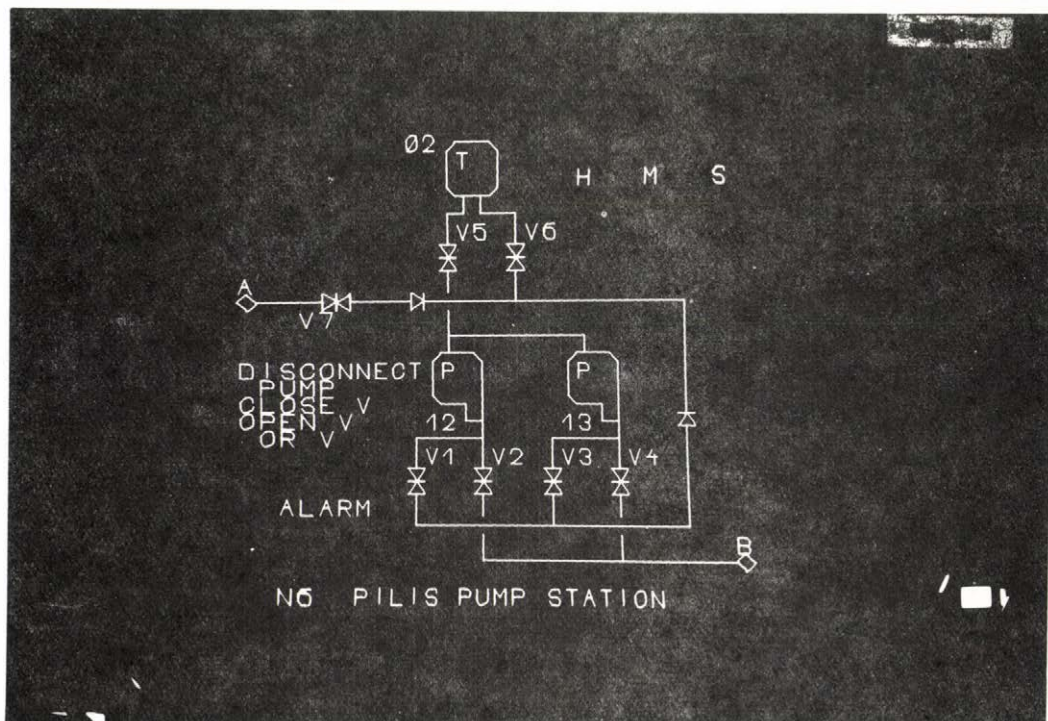
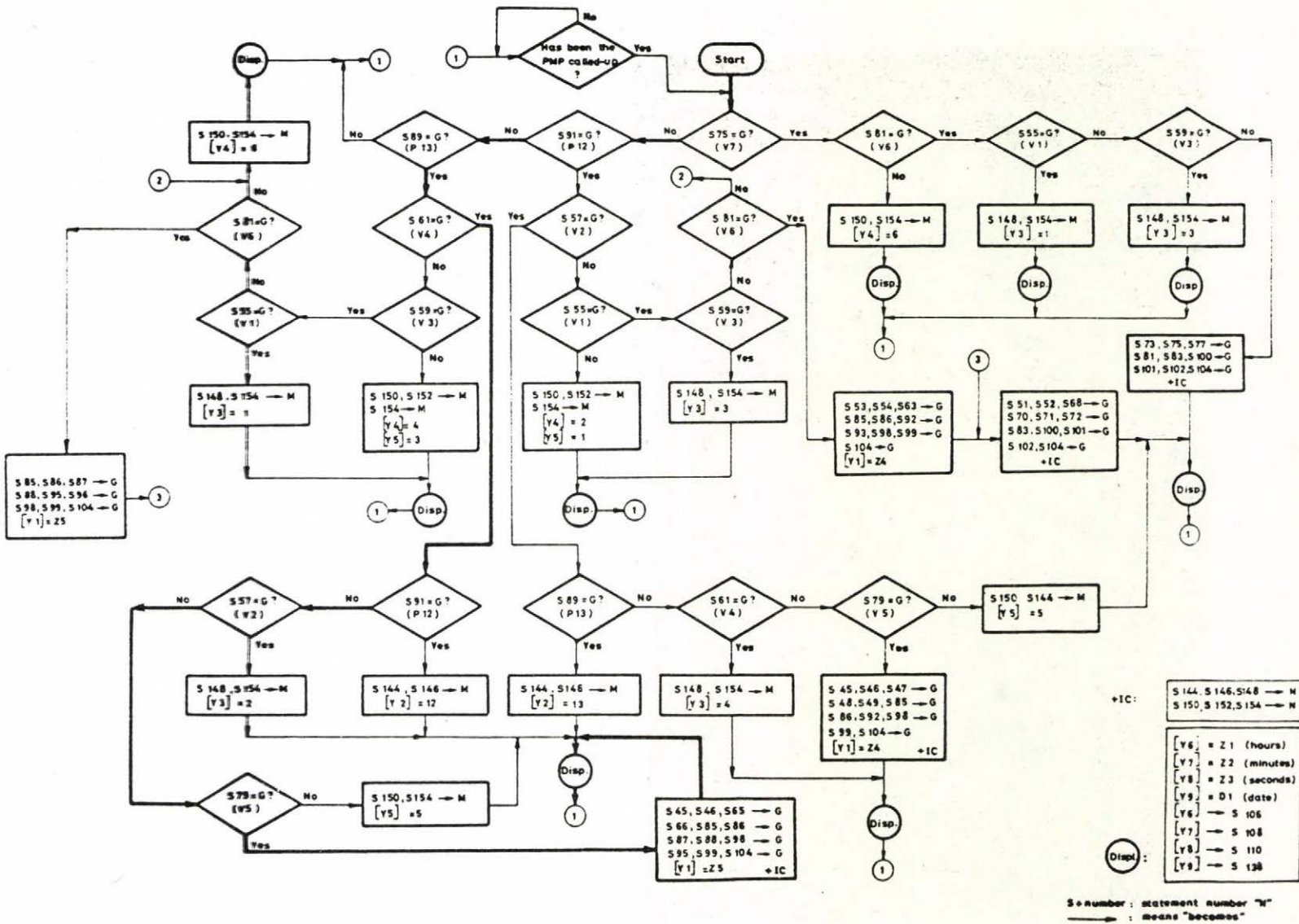


Fig. 4.7

Fig. 4.9



That is the reason why the "Call to Subroutine" statements appear without their corresponding color parameter adequately specified, and the special type of local variable [C] has not been used in statements of the graphical subroutine.

The meaning of "Names" selected for the graphical elements used is the following:

- HL: horizontal line
- VL: vertical line
- CROS: cross symbol
- ROMB: diamond /or rhombus/ symbol
- DIOD1: symbol of a diode oriented to the right
- DIOD2: symbol of a diode oriented to the bottom
- DIOD3: symbol of a diode oriented to the left
- DIOD4: symbol of a diode oriented to the top

Numbers used to indicate the modifiers /i.e. type of cuts/ appearing in the Lists, correspond with those shown in Fig. 2.22 and Fig. 2.23 /See Sect. 2.5.3.4/.

It can be seen from the List of Appendix 1 and from the Fig. 4.8, that there have been used in this example, in total:

- 155 instructions of the Graphical Language
- 284 computer memory locations in the Picture File
- 290 cells of the useful surface,

when the graphical subroutines are considered as a part of the Virgin Program.

If we define the data compression obtained as the relation:

$$DC = 2 \frac{NC}{ML}$$

where: DC is the data compression obtained for the picture

NC is the number of cells of the useful surface
"S" required to represent the picture

ML is the number of computer memory locations re-
quired to store the Virgin Program,

then we have obtained in this example a data compression:

$$DC = \frac{2 \times 290}{284} = 2.04 \text{ cells/memory location}$$

If the graphical subroutines are not included within the
Virgin Program, then there would be required:

I = 112 instructions

ML = 216 computer memory locations in the Picture
File

NC = 290 cells of the useful surface.

In that case the data compression obtained is:

$$DC = \frac{2 \times 290}{216} = 2.68 \text{ cells/memory location}$$

which is greater than the data compression obtained when
the position information is explicitly stored and one com-
puter memory location is used to store the information re-
lated to each cell of the useful surface. In that case, the
data compression will always be unity.

In this example the meaning given to local variables and
data, and the numerical value acquired each time by them is:

- [Y1] flux rate of the pump which is functioning
 /2 digits/
 [Y1] = Z4 /for the pump P12/
 [Y1] = Z5 /for the pump P13/
[Y2] number of the pump which is functioning
 /2 digits/
[Y3] number of the valve to be closed /1 digit/
[Y4] number of the first valve to be opened
 /1 digit/
[Y5] number of the second valve to be opened
 /1 digit/
[Y6] = Z1 hours /2 digits/
[Y7] = Z2 minutes /2 digits/
[Y8] = Z3 seconds /2 digits/
[Y9] = D1 date /12 characters/

CHAPTER 5

SIMULATION OF THE GENERAL CONCEPT PROPOSED FOR THE DISPLAY SYSTEM

5.1 Introduction

In the present study we have analyzed several topics in connection with the design and the general concept of an interactive cell-organized color raster-scan graphical display system. As a result, several criteria have been established which take into account some conclusions at which we have arrived, in order to get the best solution for a graphical display system oriented to process control applications. Considering the graphical display units as a part of the whole Supervision System, a manipulating system has also been proposed. Finally, a special-purpose Graphical Language has been designed to create the drawings required by the application, which applies our criteria in some related topics and takes into account either some requirements established by us, or those particularly imposed by the application itself.

5.2 Objectives of the Simulation

This Chapter deals with the results obtained by using the Graphical Language in creating several drawings, by means of which the supervision and control of industrial process, the preparation of economic reports, statistical records, etc., can be carried out. Thus, the Simulation realized shows the advantages attained with statements proposed and the validity of the criteria established in the present work, proves the efficiency and flexibility of the statements created for the Graphical Language, the data compression obtained with them, and in some measure proves also the realizability of

the whole Manipulating System.

5.3 Equipment used

The Simulation has been carried out using a TPA-70/1025 minicomputer and a GD'71 monochromic random-scan Graphical Display, both of Hungarian manufacture. The Disk Operating System of the TPA /DOST/, as well as the whole configuration operating in the Computer and Automation Institute of the Hungarian Academy of Sciences, has served these purposes very efficiently, rendering great flexibility in the preparation, debugging and handling of the Virgin Programs prepared.

5.4 Development of the task

To carry out the Simulation, an Interpreter was created by means of which the statements composing the Virgin Programs prepared were transformed to the proposed coding /See Sect. 4.4.4/. Starting from this coding, the corresponding display instructions /See Fig. 4.5/ were created following the sequence of statements in each particular Virgin Program prepared. The position information and the data information, related to each hypothetical cell created on the random-scan graphical display, were used each time to call the corresponding subroutines programmed to create the alphanumerics and symbols oriented to the application, and to settle in place the respective visual information on the screen of the random-scan graphical display. These operations were realized by a computer program which broke up the compressed graphical display instructions in single ones, and transformed and stored them in the corresponding place in the operative computer memory which serves as the refresh memory for the random-scan graphical display. A 32x32-cell grid showing the position of the cells on the screen, was also created to facilitate the debugging of Virgin Programs and to show better the principle of the cell-organized raster

display. Once the drawing was finally created, removing of the grid was possible in order to appreciate better the quality of the picture being represented and to present it in a similar form as when it would appear in an actual raster-scan display.

The color was not simulated, as a monochromic graphical display was used. Nevertheless, conditions were created in the Interpreter to identify it. In the Simulation neither the local variables, nor the data and "Z" Values were used /See Sect. 4.4.3.1/ to update the drawings remotely, so they were all shown as "static" ones. Nevertheless, they could be altered very easily by modification of the corresponding computer words in the coding created. It was possible to obtain each time, when desired, the list with statements used in the program, its corresponding coded words to which they were transformed and the relative address in the computer operative memory wherein they were stored, all these adequately numbered to facilitate debugging and editing tasks /See Appendix 1 and 2/. In these lists there appeared conveniently indicated each time the syntactical and programming errors present in the Virgin Program prepared. It was also possible to obtain for each program prepared, the corresponding coded word related to each cell containing graphical Information.

The cutting of symbols was simulated by creating a number of independent symbols with the corresponding codes for each symbol which would be obtained with the types of "cuts" proposed /See Sect. 2.5.3.4/.

In general, Virgin Programs were prepared in the following sequence:

- fixed part: this includes the invariable graphics /which may, or may not use the graphical sub-routines/ and occasionally the numerical code of the diverse components which never

varies in the design;

- legends, labels, etc.: including the invariable texts in the drawing;
- variable part: which includes texts and graphical information which varies directly with the coded information arising from the process and/or as a result of its processing.

However, according to the general concept of the Graphical Language and of the Manipulating System proposed, any other arbitrary sequence can be followed.

The time required to compile and effectuate the required transformations and decomposition of compressed graphical instructions, was never longer than 4 seconds for the most complex program prepared. On the other hand, the time required on average to draw a drawing, to prepare its corresponding Virgin Program and to debug it, was approximately 6 hours. The time consumed in typing the statements has obviously not been considered.

5.5 Virgin Programs prepared

The enumeration of some of the Virgin Programs prepared in the Simulation is given in the Table 5.5.1. For each one are indicated, according to the definition given in Section 4.5, the data compression obtained when the graphical subroutines are considered as a part of the Picture File wherein the Virgin Program is stored /case 1/, and the data compression obtained when they are considered as common to all the Virgin Programs created for the system /case 2/. Finally, Table 5.5.1 shows also the number of the Figure of the corresponding picture displayed on the graphical display and created by means of each Virgin Program prepared. Each row also indicates the application to which each drawing is

Table 5.5.1

Picture	Name of Virgin Program	<u>Data Compression</u>		Fig. No.	Observations
		Case 1	Case 2		
1	SOURCE	2.41	2.59	5.2	Character and Symbol-set used
2	TABLE	5.10	5.68	5.3	Construction of tables, lists, etc.
3	BAR DIAGRAM	6.24	8.20	5.4	Example of an Economic Report
4	PUMP STATION	2.00	2.80	5.5	Application in Pipelines and Bulk Storage
5	MONTHLY RECORD	4.47	5.44	5.6	Example of a Statistical Record
6	ELECTRIC STATION	2.60	3.10	5.7	Application in Electric Power Distribution Networks
7	DISTILLATION COLUMN	1.62	3.36	5.8	Application in Chemical and Petroleum Processing
8	ELECTRIC SUBSTATION	2.00	3.00	5.9	Application in Electric Power Distribution Networks
9	EVAPORATOR CONTROL	2.20	4.32	5.10	Application in Sugar Factories
10	SULPHITATION STATION	2.41	4.10	5.11	Application in Sugar Factories

related.

Obviously, the values of data compression given in Table 5.5.1 indicate only the approximate order which can be obtained, because in each case it would depend on the particular program prepared. In any case it cannot be considered that the optimal program has been prepared, since they were arbitrarily selected each time from many possible solutions. These values of data compression for the Virgin Programs are greater if data and local variables would be specified in statements of the graphical subroutines. In this case an additional saving is obtained, since some statements do not need to be specified when the same graphical subroutine is used two or more times in the Virgin Program.

In Fig. 5.4 to Fig. 5.11 the cells, instructions, etc. required to create the Table included within each drawing, have not been considered in the calculation of the values of the data compression given in Table 5.5.1. They represent the total number of statements used in the Virgin Program /I/, the number of memory locations required to store the coding created /ML/, the number of cells being used to represent the drawing /NC/, the data compression in Case 1 /DC/ and the data compression in Case 2 /T/.

Other additional applications to those enumerated in Table 5.5.1 are:

- As color alphanumeric displays
- Railways Traffic Control and Transport Networks in general
- Financial and Business Environment
- Medical Monitoring Systems
- Creating Organigrams, Flow Diagrams, Karnaugh maps, etc.
- In creating group parlor games
- In limited artistic applications

5.6 Experiences obtained with the Simulation

The most important experiences obtained with the Simulation have been the following:

1. The use of the slanted bar "/" instead of circular brackets in the syntax of statements should diminish the time required for typing the Virgin Programs, because it is not necessary to distinguish between the two possible types of parenthesis "(" and ")", and it does not require each time the use of the "shift" key. Moreover, because each type of statement is distinguished from the remainder by the different initial letter assigned, the use of the slanted bar could be avoided without giving place to ambiguities or lack of definition.
2. As far as possible, it is advantageous to use the greatest possible number of graphical subroutines, i.e. not only for the definition of complex symbols, but also for topologically equal parts in a drawing.
3. It is more efficient from the point of view of the number of computer locations required, to use the "Move without repetition" statements instead of "space" characters specified in "Text" statements, in those cases when several alphanumerics are not successively arranged in the same column or row. This is particularly significant when the number of "space" characters required is greater than 2.
4. The use of "Move with repetition" statements following a "Call to Subroutine" statement is advantageous only in those cases when the particular subroutine does not change the color specified in the statement. In any other case it is convenient to separate them, that is, to use an independent statement for each equal graphical subroutine.

5. It would be advantageous to designate a single color for every single horizontal Text for every drawing, since it is the most common demand when alphanumeric information is specified. This means to substitute the orientation, color and repetition parameters by a particular letter in Text statements. Alternatively, an even better solution could be to permit the use of either, the color specification in "Text" statements only, or the complete specification of all the parameters. In other words, to create conditions in the Interpreter which permit us to consider as equivalent the following possibilities in "Text" statements:

T (+H,B,1) SOL and T B SOL
T (+H,M,1) LUNA and T M LUNA
T (+H,Y,1) ASTRO and T Y ASTRO

6. If a horizontal or vertical line /or in any other case when a single symbol is repeated several times in one direction/ is intersected by more than two or three other single symbols, it is generally equivalent from the point of view of the total number of statements required, to create first the straight line and finally to superpose over it the single symbols required. /1st. solution/, or to create it by sections, that is, as it appears in the drawing /2nd solution/. These alternatives are graphically equivalents, but the first solution requires less computer words in the structure of the Virgin Program than the second one.

As an example, suppose that a straight line occupying 13 cells is intersected with contact 3 times /Fig. 5.1./ The sequence of the types of statements required in case it is traced by sections /2nd solution/ will be: $M_1, E_1, E_2, E_3, E_4, E_5, E_6, E_7, M_2$; in this case, 7 "Element" and 2 "Move without repetition" statements /i.e. 9 statements/ and 16 memory locations /See Table 4.4.3/ are required for the

13 cells. If the symbols for the intersection are superposed after the straight line has been created /1st solution/, then the sequence of statements will be: $M_1, E_1, M_2, E_2, M_3, E_3, M_4, E_4, M_5$; in this case, 4 "Element" and 5 "Move without repetition" statements, /i.e. 9 statements also/, but only 13 memory locations are required for 13 cells on the useful surface.

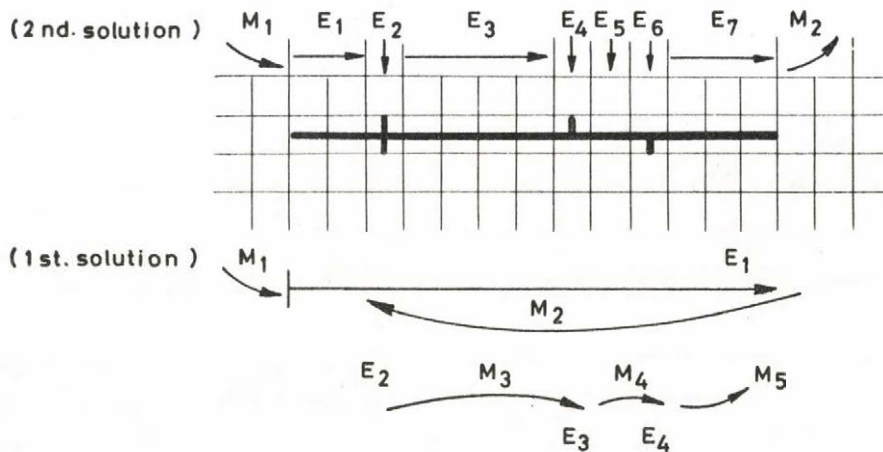


Fig. 5.1

Nevertheless, the first alternative can not be used in every case, because it does not permit the updating /i.e. change in color, for instance/ of each segment separately; using the second alternative this inconvenience is not present. Therefore, it is commendable to use the second solution in networks of any type, and the first one when the situation illustrated takes place as a part of a complex symbol, in whose case all the single symbols require to be always changed to the same color simultaneously.

7. The repetition facility in "Move with repetition" statements does not need to be, in general, very big in process control applications, 32 repetitions being sufficient /instead of 64/, in which case only 5 information bits are required.

8. The use of graphical subroutines to define complex symbols occupying a small number of cells is only justified when they are required many times in a given drawing. On the other hand, they must be used for those complex symbols which occupy many cells on the useful surface, even if they need to be represented only twice in a drawing.
9. The use of Data and Local Variables increases the data compression in Virgin Programs mainly in drawings composed of several repetitive parts.
10. It is possible to create straight lines of any length on the useful surface using a maximum of 2 instructions, i.e. an "Element" instruction followed by a "Move with repetition" instruction.
11. The types of statements proposed for the Graphical Language, give great flexibility in creating the drawings and permit us to construct them in a modular form. Additionally, they are easily memorized and therefore they permit us to create and debug the Virgin Programs in a very straightforward form.
12. In many cases, complex symbols can be created starting from other complex symbols, e.g. the symbol for the pump is created from the symbol designed for a tank. This permits us to save even more on computer locations and facilitates the programming task.
13. It would be commendable in listing the Virgin Programs, to assign a special editing symbol to indicate the presence of "space" characters when used in "Text" statements, mainly in those cases when they are used with other objectives than to separate the words of the text itself. The editing symbol "⌞" could serve for this objective.

14. In "Element" and "Text" statements, the "repetition" parameter "r" represents exactly the number of times a given single symbol must be repeated. On the other hand, the repetition "R" in "Move with repetition" statements indicates the additional number of times the previous graphical statement must be repeated, after it has been created once. This situation could give rise to errors mainly when the programmer is inexperienced.

5.7 Conclusions for the present study

The Simulation of the general concept proposed for the display system to be used in process control applications has demonstrated that:

1. It is realizable.
2. The Graphical Language /Chapter 4/ wholly fulfills the objectives for which it was created.
3. A considerable data compression is obtained with the types of statements /Sect. 4.4/ and the coding proposed /Sect. 4.4.4/, offering also an easy means to update the drawings.
4. The Virgin Programs can be prepared very easily, with great flexibility, and without requiring special skills.
5. The selection of the requirements for the drawings /Sect. 2.5/, of the criteria for the symbols oriented to the application /Sect. 2.5.2/ and of the design criteria for the display units /Sect. 3.6.1.1/, fits into the actual demands of a real display system to be used in the supervision and control of industrial processes.
6. The use of only four information bits to specify the repetition of single symbols /Sect. 2.6.1/ is sufficient, permitting us to get the flexibility required to construct the drawings required, to save information bits in the instruction-words and to use a minimal number

of statements in creating the straight lines demanded by the application.

7. Good use is made of the 16-bit computer words to store the information required by every single graphical element /including the color/, with 16 cells as a maximum /Sect. 2.6.3/.
8. The use of square cells /Sect. 2.4/ has advantages over other rectangular cells, mainly due to its possibility of coupling in all directions.
9. The use of the principle of cutting a given number of well-selected single symbols /Sect. 2.5.3.4/ permits us to get pictures with good quality, to give a better use to the ROM /or PROM/ required, and to "multiply" the whole Symbol-set created for the application.
10. The general concept proposed for the whole display system is consistent with the requirements imposed and propositions given for the Manipulating System described in this study /Sect. 3.6/.

BETUK		ABCDEFGHIJ							
		KLMNOPQRST							
		UVWXYZ							
SZAMOK		0123456789							
SZIMBOLUMOK		KIVAGAS							
1	HL	—							
2	VL	I							
3	FC	■	0	1	2	3	4	5	6
4	CROS	+	0	1	2	3	4	5	6
5	ROMB	◇	0	1	2	3	4	5	6
6	D	0D1	1	2	3	4	5	6	7
7	D	0D2	1	2	3	4	5	6	7
8	D	0D3	1	2	3	4	5	6	7
9	D	0D4	1	2	3	4	5	6	7

Fig. 5.2

NAPI INPUT X1000 FT			
	ALMA	KORTE	SZILVA
12 40			
15 00			
8 10			
26 76	NC	193	DC 5 10

Fig. 5.3

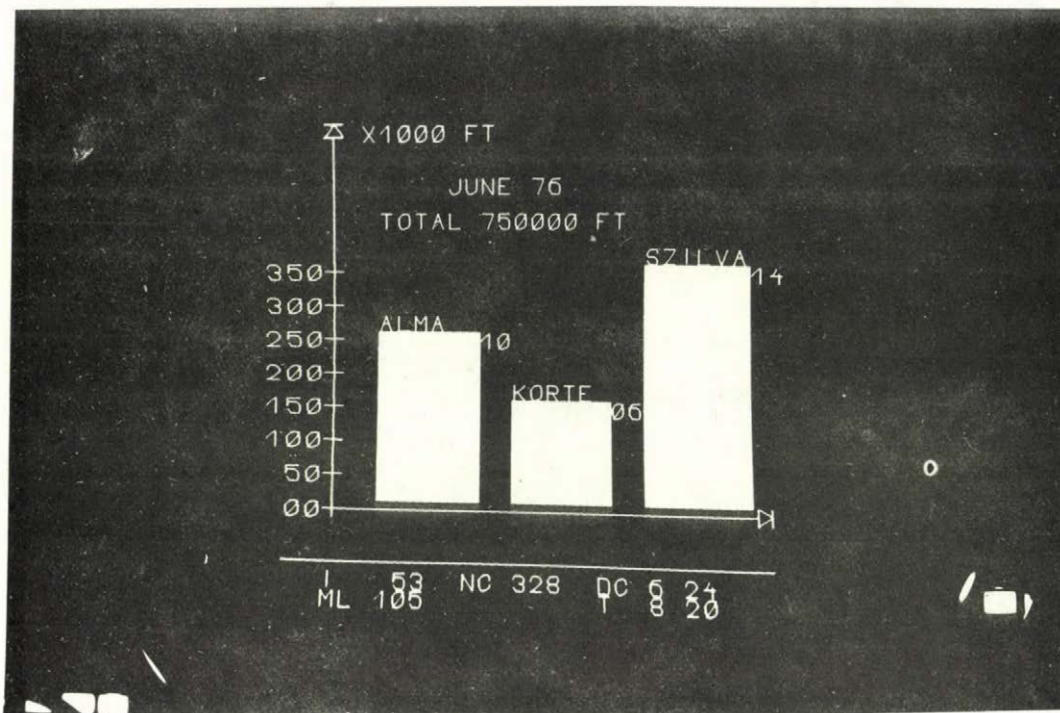


Fig. 5.4

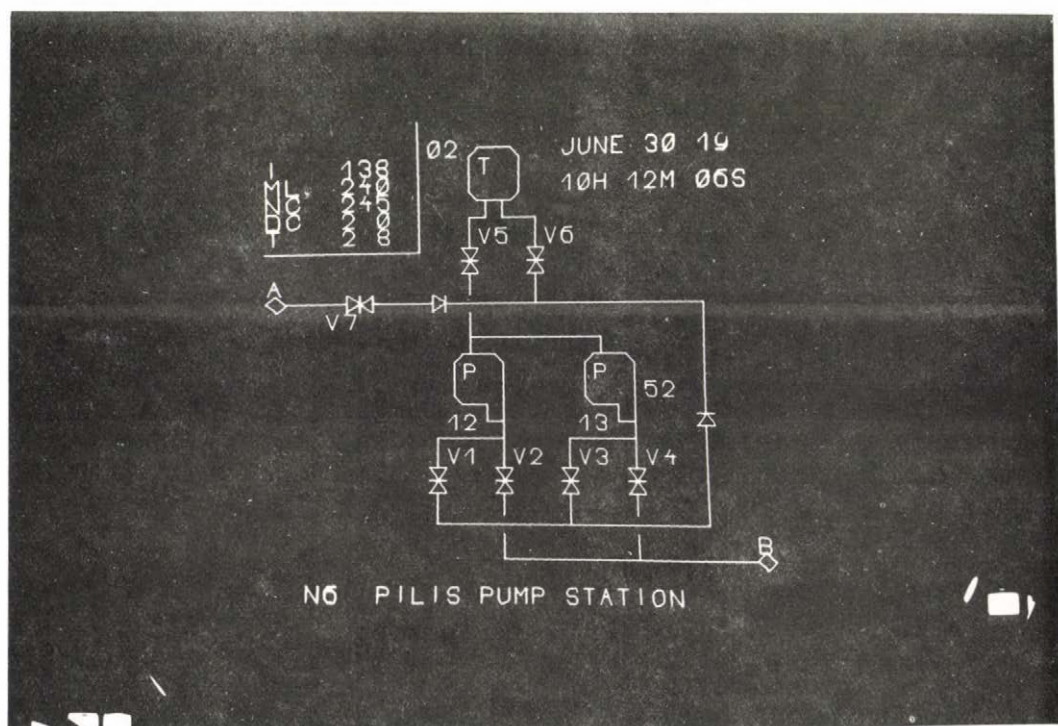


Fig. 5.5

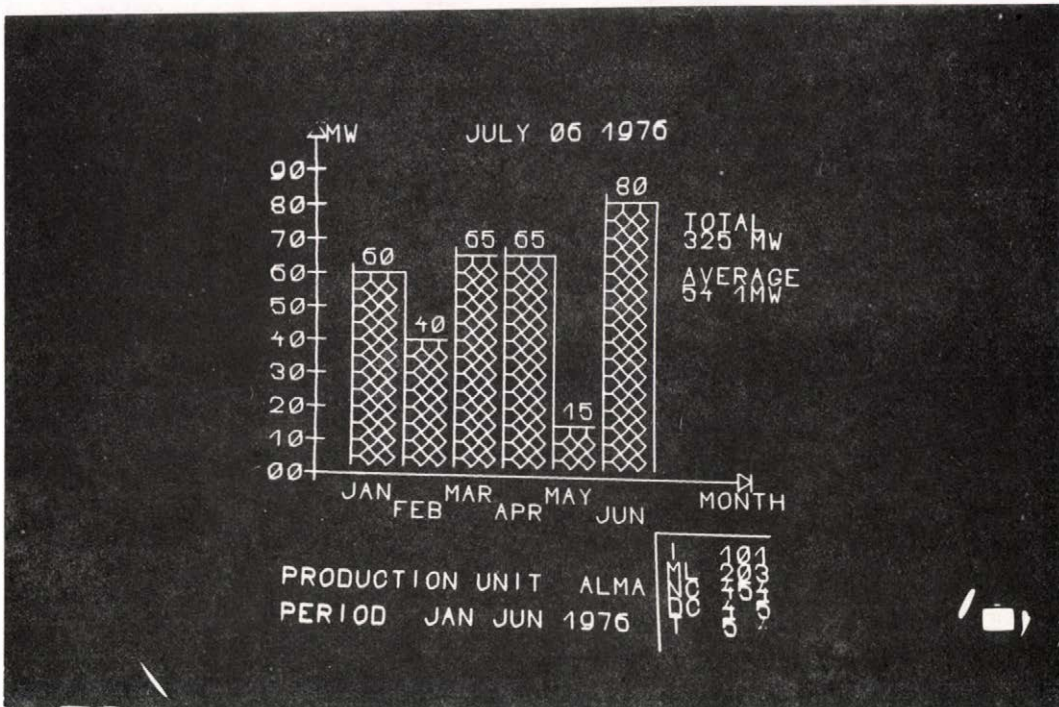


Fig. 5.6

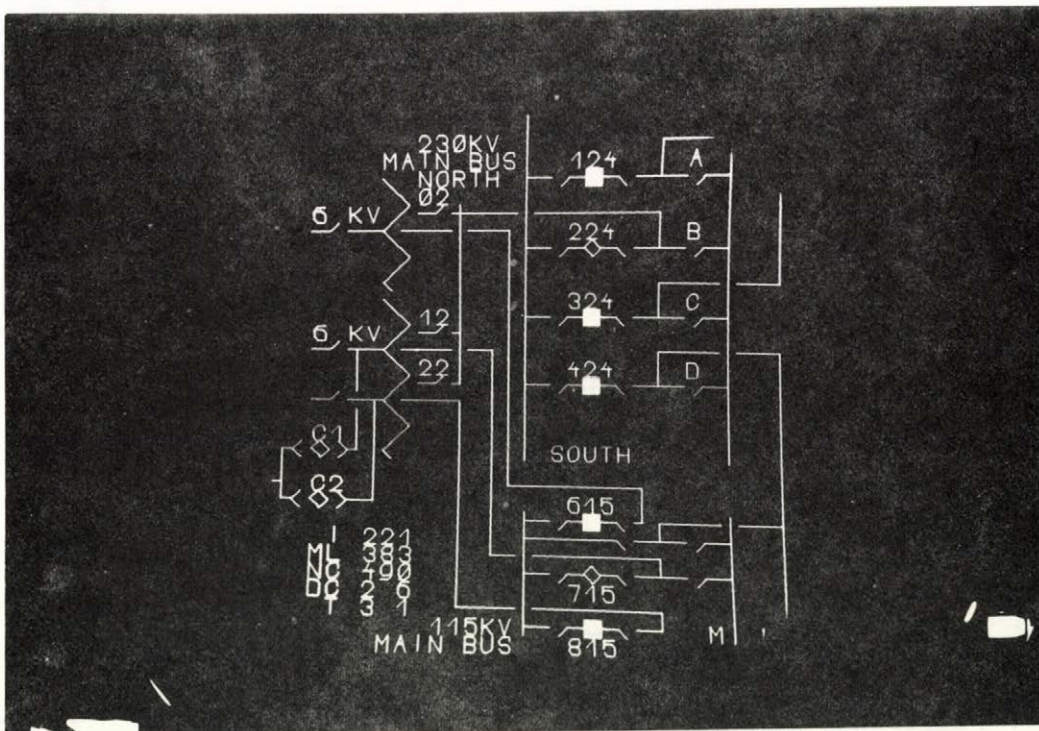


Fig. 5.7

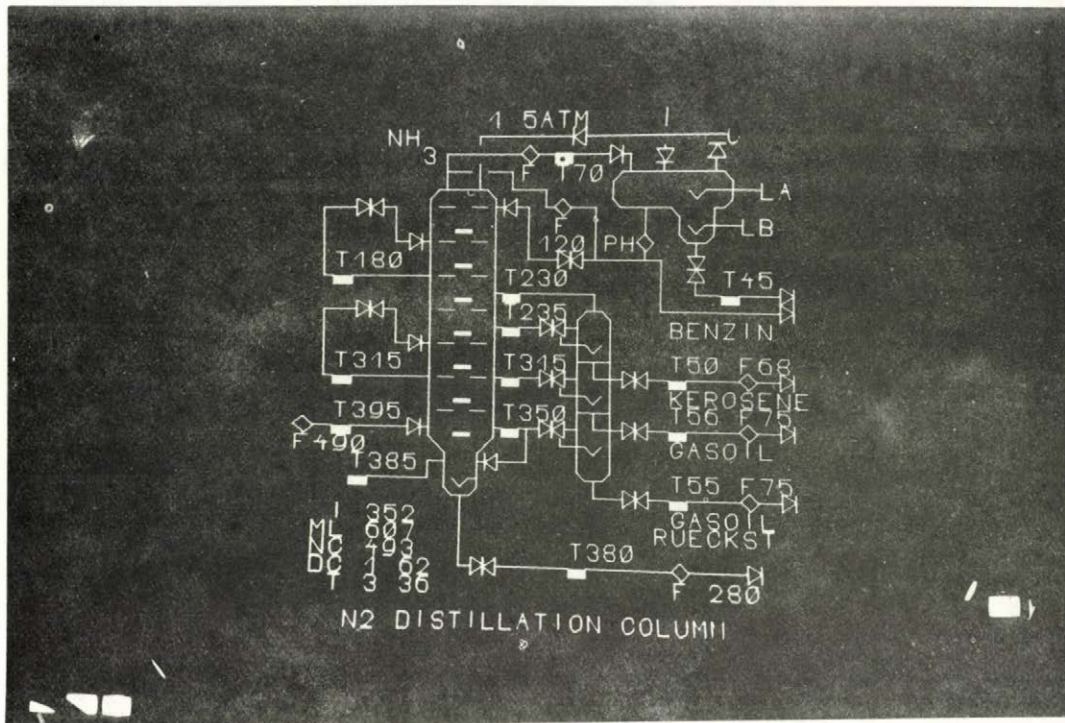


Fig. 5.8

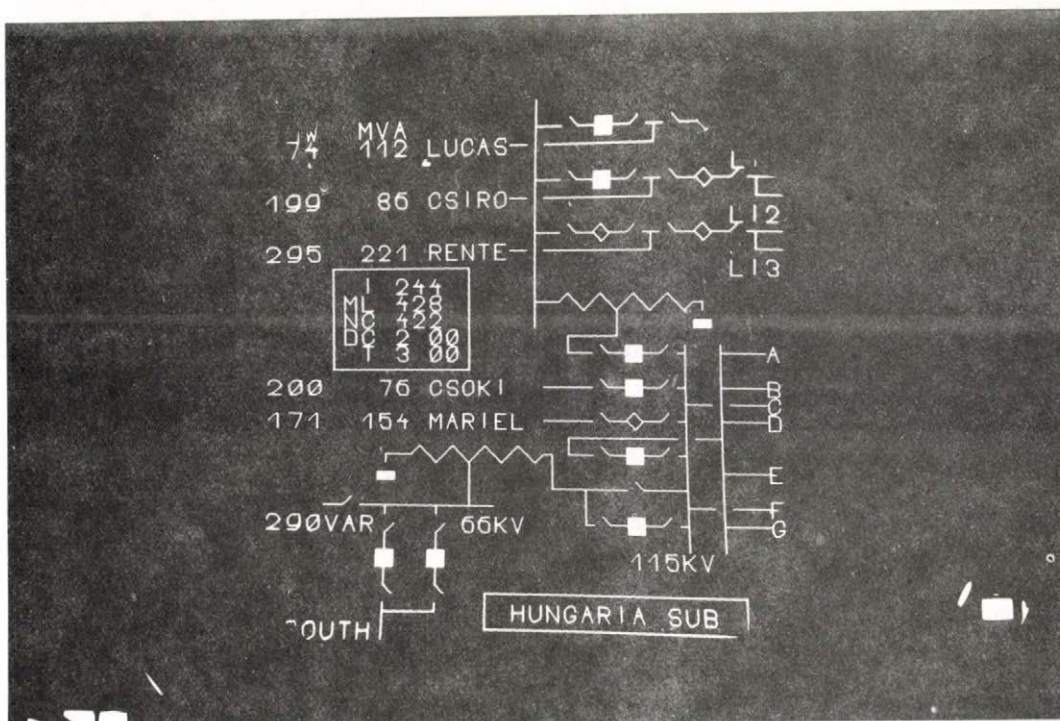


Fig. 5.9

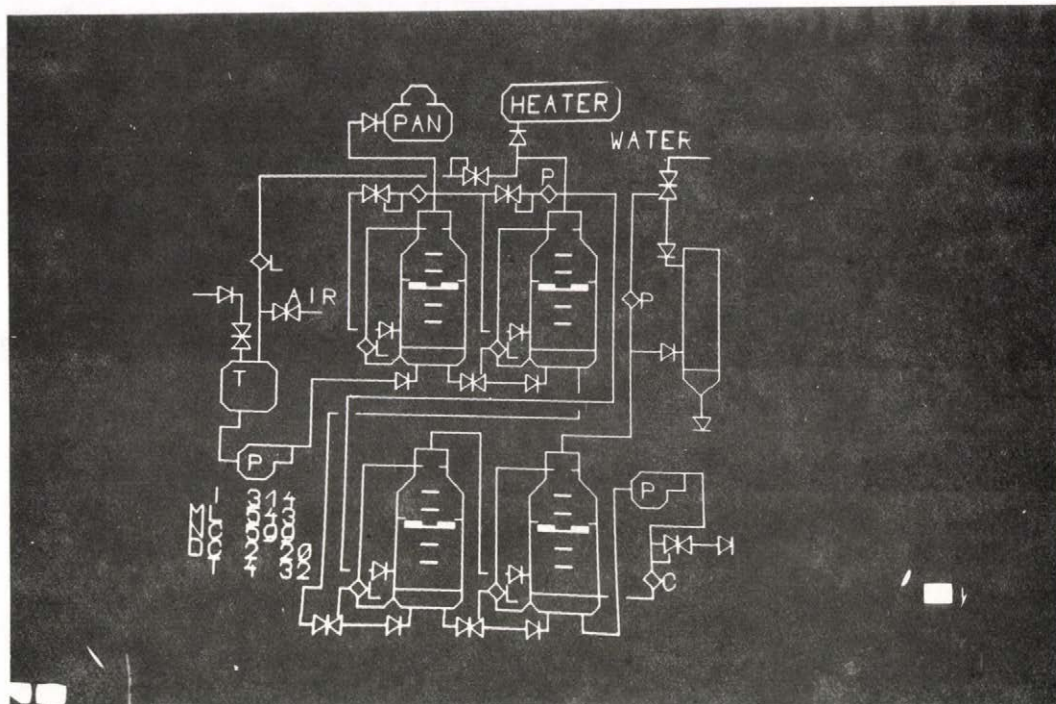


Fig. 5.10

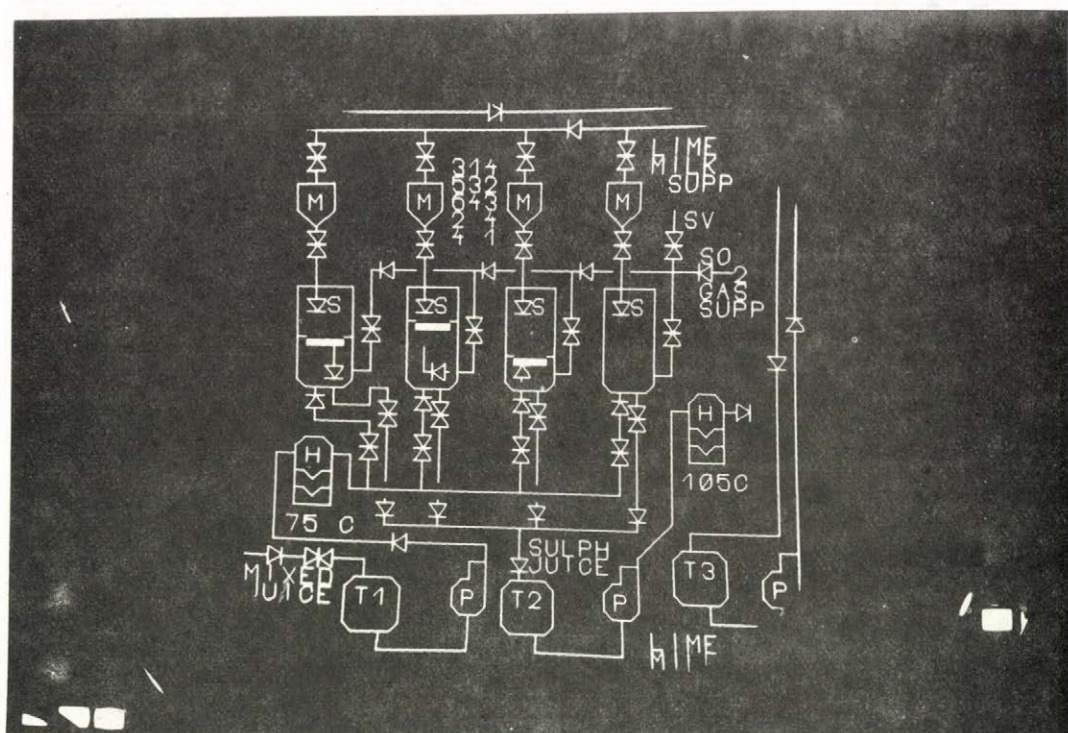


Fig. 5.11

REFERENCES

- [1] Adcox, T.S.: "Alphanumeric CRT terminals", The Electronic Engineer, Nov. 1971, pp. DC9-11.
- [2] Agajanian, A.H.: "A bibliography on display technologies", Proceedings of the SID, Vol.14, No.2, 2nd Quarter 1973, pp. 76-80.
- [3] Altman, L.: "Special Report: The medium of the message will soon be optoelectronic", Electronics, Vol.45, No.4, Febr. 14/1972, pp. 60-71.
- [4] Altman, L.: "Special Report: Semiconductor RAMs land computer mainframe jobs", Electronics, Vol.45, No.18, Aug. 28/1972, pp. 63-77.
- [5] Altman, L.: "Semiconductor random-access memories", Electronics, Vol.47, No.12, June 13/1974, pp. 108-110.
- [6] Altman, L.: "Charge-coupled devices move in on memories and analog signal processing", Electronics, Vol.47, No.16, Aug. 8/1974, pp. 91-101.
- [7] Anderloni, A.; Bisci, D.; Durazzo, M. et al.: "CRT displays with reference to their use in process control", Elettrotecnica, Vol.62, No.11, Nov. 1975, pp. 957-970.
- [8] Armstrong, J.R.; Hern, C.L.: "Convert your scope to a display terminal", Electronic Design 23, Nov. 11/1971, pp. C20-24.
- [9] Bachman, C.W.: "The evolution of storage structures", Communic. of the ACM, Vol.15, No.7, July 1972, pp. 628-634.

- [10] Bainbridge, L.: "The influence of display type on decision making", IEE Conference on Displays, 7-10 Sept./71, Loughborough, England, pp. 209-215.
- [11] Baron, S.N.: "A Television Compatible Character Generator", Information Display, Vol.9, No.4, July-Aug./1972, pp. 13-16,30.
- [12] Barquin, R.C.: "The state of computation in Cuba", Datamation, Dec. 1973, pp. 69-72.
- [13] Barrett, R.C.; Jordan, B.W.: "Scan Conversion Algorithms for a Cell Organized Raster Display", Communic. of the ACM, Vol.17, No.3, March 1974, pp. 157-163.
- [14] Beck, M.S.; Wainwright, N.: "Direct Digital Control of Chemical Processes", Control, Sept. 1967, pp. 428-432.
- [15] Beck, M.S.; Wainwright, N.: "Direct Digital Control of Chemical Processes", Control, Oct. 1967, pp. 508-510.
- [16] Beck, M.S.; Wainwright, N.: "Direct Digital Control of Chemical Processes", Control, Nov. 1967, pp. 547-550.
- [17] Beck, M.S.; Wainwright, N.: "Direct Digital Control of Chemical Processes", Control, Dec. 1967, pp. 593-595.
- [18] Beck, M.S.; Wainwright, N.: "Direct Digital Control of Chemical Processes", Control, Jan. 1968, pp. 53-56.
- [19] Benson, D. et al.: "A language for real-time systems", The Computer Bulletin, Vol.11, No.3, Dec. 1967, pp. 202-212.
- [20] Berényi, I.: "USSR aims at big computer by '75, Electronics, Vol.45, No.20, Sept. 25/1972, pp. 72-75.

- [21] Berkeley, E.C.; Bobrow, D.G. /Editors/: "The Programming Language LISP: Its Operation and Applications", Information International, Maynard, The M.I.T. Press, Massachusetts, 1964.
- [22] Berkowitz, S.: "PIRL-Pattern Information Retrieval Language-Design of Syntax", Proceedings of 1971 Annual Conf. ACM, Aug. 1971, pp. 496-507.
- [23] Bernát, I.; Ladányi, L.: "Incremental Methods for Image Generation in CRT Display Systems", Technical Paper A42-7/1, Hung. Acad. Sci., Comp. Aut. Inst., Budapest, 19 pp.
- [24] Berndt, H.: "Codefamilien", Data Report 5, 1970, Heft 2, pp. 18-23.
- [25] Biri, J.; Lukács, J.; Somlai, L.; Vashegyi Gy.: "Industrial Data Logging System", Proceedings of the 1st International Symposium on CAMAC, Issue 9, April 1974, pp. 269-271.
- [26] Bishop, P.G.: "Display and input software for on-line control", IEE Conference on Displays, 7-10 Sept./71, Loughborough, England, pp. 19-26.
- [27] Bjelland, H.: "Display articles - indexed from current periodicals", Proceedings of the SID, Vol.13/1, First Quarter 1972, pp. 67-86.
- [28] Bjelland, H.: "Display articles - indexed from current periodicals", Proceedings of the SID, Vol.13/2, Second Quarter 1972, pp. 125-139.
- [29] Blackwell, F.W.: "An on-line symbol manipulation system", Proceedings of 22nd Nat.Conf. ACM, 1967, pp. 203-209.
- [30] Boardman, T.L.: "Hardware/software design considerations for high speed/low cost interactive graphic communication systems", AFIPS, National Computer Conf. 1974, Proceedings, Vol.43, pp. 273-278.

- [31] Boas, S.C.: "Visual Display Terminals for Computer-controlled Systems", Post Office Electrical Eng. Journal, Vol.65/2, July 1972, pp. 108-114.
- [32] Borrow, D.G.; Raphael, B.: "A Comparison of List-Processing Computer Languages", Communic. of the ACM, Vol.7, No.4, April 1964, pp. 231-240.
- [33] Boyd, S.H.: "Digital-to-Visible Character Generators", Electro Technology, Jan. 1965, pp. 77-88.
- [34] Boyland, D.A.; Norman, D.J.; Allard, L.S.: "Bi-Colour Display Tubes", IEE Conference on Displays, 7-10 Sept/71, Loughborough, England, pp. 39-47.
- [35] Bratt, B.: "TV-set is display for data Terminal", Electronic Design 19, Sept. 14/1972, pp. 134-141.
- [36] Braun, J.E.; Stewart, H.G.: "The Graphic Display at an Electrical Power Interconnection Control Center", SID Journal, Vol.1, No.2, July-Aug./1972, pp. 16-23,27.
- [37] Brown, G.H.: "Television's role in Tomorrow's world", IEEE Spectrum, Oct. 1967, pp. 56-58.
- [38] Brown, J.R.: "First-and Second-order Ferrite Memory Core Characteristics and their Relationship to System Performance", IEEE Transactions on Electronic Computers EC-15, Aug. 1966, pp. 485-501.
- [39] Bryden, J.E.: "Design Considerations for Computer-Driven CRT Displays", Computer Design, Vol.8, No.3, March 1969, pp. 38-46.
- [40] Bryden, J.E.: "Visual Displays for Computers", Computer Design, Vol.10, No.10, Oct. 1971, pp. 55-77.
- [41] Bryden, J.E.: "Some notes on measuring performance of Phosphors used in CRT displays", SID 7th National Symp. on Information Display, Oct. 1966. pp. 83-103.

- [42] Burns, R.; Savitt, D.: "Microprogramming, stack architecture ease minicomputer programmer's burden", Electronics, Vol.46, No.4, Febr. 15/1973, pp. 95-101.
- [43] Caplan, D.I.: "Software/Hardware Tradeoffs in the Design of an Intelligent Display Terminal", SID, Int. Symp. Digest of Technical Papers 1974, Vol.V, pp. 64-65.
- [44] Childress, L.S.: "Selecting the proper Display System", Information Display, May-June/1971, pp. 19-23, 51.
- [45] Christensen, C.: "Examples of Symbol Manipulation in the AMBIT Programming Language", Proceedings of the ACM, 20th Nat. Conf. 1965, P-65, pp. 247-261.
- [46] Christensen, C.; Shaw, C.J. /Editors/: "Seven Extensible Languages: GPL, ALGOL 68, Basel, Imp. PPL, Proteus, EPS", Extensible Languages Symp., Boston 1969, ACM SIGPLAN Notices, Vol.4, No.8, Aug. 1969, 62 pp.
- [47] Clarke, J.; Welbourne, D.: "Display Systems for use on-line in Power Stations", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 125-132.
- [48] Clauer, C.K.; Neal, A.S.; Erdmann, R.L.: "Evaluations of some display parameters with human performance measures", SID, 7th National Symposium of Inf. Disp., Boston Mass., Oct. 1966, pp. 15-23.
- [49] Coggan, B.B.: "The Design of a Graphic Display System", Report No.67-36, Univ. of California, Aug. 1967, 192 pp.
- [50] Cohen, D.; Lee T.M.P.: "Fast drawing of Curves for Computer Display", Proceedings of the 1969 Spring Joint Computer Conference, pp. 297-307.

- [51] Cole, C.F.: "Cyclic Codes in Analog-to-Digital Encoders", Computer Design, Vol.10, No.5, May 1971, pp. 107-112.
- [52] Conde, C.A.N.; Correia, C.A.; Figueiredo, A.D.: "Register Addressing System Access within Nanoseconds", Electronics, Vol.46, No.24, Nov. 22/73, pp. 117.
- [53] Conn, A.P.: "GRIND: a Language and Translator for Computer Graphics", Technical Report, Thayer School of Engineering, Hanover, New Hampshire, June 1969, 82 pp.
- [54] Copping, B.; Alexander, V.D.; Hunter, J.J.: "Human Factor Assessment of Some Numeric Visual Displays", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 27-32.
- [55] Cotton, I.W.; Greator, F.S.: "Data Structures and techniques for remote computer graphics", Proceedings 1969 Fall Joint Computer Conf. AFIPS, Vol.35, Nov. 1969, pp. 533-544.
- [56] Curtis, D.A.: "Mass Memory Technologies Part 1", Instrument and Control Systems, Oct. 1971, pp. 107-109.
- [57] Dallimonti, R.: "The human operator in process control systems", European Conf. on Electro-technics, EUROCON'74 Digest, Amsterdam, Netherland, 22-26 April/1974, A3-6/2 pp.
- [58] Damon, P.: "High Resolution multi-color storage tube", Information Display, Nov.-Dec./1966, pp. 31-35.
- [59] Davis, S.: "Disc Storage for Minicomputer Applications", Computer Design, Vol.12, No.6, June 1973, pp. 55-66.

- [60] Davis, S.: "A Fresh View of Mini- and Microcomputers",
Computer Design, Vol.13, No.5, May 1974, pp. 67-79.
- [61] Davis, S.: "Selection and Application of Semiconductor
Memories", Computer Design, Vol.13, No.1,
Jan. 1974, pp. 65-77.
- [62] Davis, S.: "A Comparison and Review of Digital
Readouts", Computer Design, Vol.11, No.3,
March 1972, pp. 69-81.
- [63] Di Giulio, H.A.; Tuan, P.L.: "A graph manipulator for
on-line network picture processing", Proceedings
AFIPS, FJCC, 1969, pp. 387-398.
- [64] Dimo, P.D.; Gayraud, A.J.; Popovici, N.N.: "A flexible
low cost Alphanumeric Display", Computer Design,
Vol.12, No.6, June 1973, pp. 77-82.
- [65] Dipotet, P.: "Algunas consideraciones sobre los
sistemas de adquisición de datos", Control,
Cibernética y Automatización, Oct.-Dic./74, Vol.8,
No.4, pp. 3-10.
- [66] Edwards, E.; Lees, F.P.: "Information Display in
Process Control", IEE Conf. on Displays, 7-10
Sept/71, Loughborough, England, pp. 133-137.
- [67] Ella, I.J.: "Alphanumerical character generation",
Computer Technology IEE, Conf. Publication,
No.32, July 1967, pp. 222-229.
- [68] Elliott, I.R.: "The use of Display Consoles in a
File Updating Scheme", The Computer Bulletin,
Vol.13, No.6, June 1969, pp. 187-189.
- [69] Ellis, A.B.E.: "A Direct View Storage Tube with
Selective Erasure as a Data Terminal Display",
IEE Conf. on Displays, 7-10 Sept/71,
Loughborough, England, pp. 1-5.

- [70] Elms, D.: "A Simplified graphics language for mini-computers", Information Display, Vol.9, No.6, Nov.-Dec./1972, pp. 9-13.
- [71] Evans, F.C.: "Generalized Karnaugh Maps", The Radio and Electronic Eng., Vol.42, No.1, Jan.1972, pp. 28-30.
- [72] Evans, S.J.W.; Cropper A.G.: "Ergonomics and Computer Display Design", The Computer Bulletin, Vol.12, No.3, July 1968, pp. 94-98.
- [73] Faubert, D.B.; Leonard, K.C.: "Real-time Multisensor Data Display", National Aerospace Electronic Conference, 1970 Record, pp. 88-95.
- [74] Felipe, E.: "Specification Problems of a Process Control Display", Report 37/1975, Hung. Acad. Sci., Comp. Aut. Inst., Budapest, 1975, 43 pp.
- [75] Fidirich I., Uzsoy M.: "LIDI-72 listakezelő rendszer a Digitális Osztályon, 1972 évi változat", MTA SzTAKI Tanulmányok 16/1974.
- [76] Flynn, M.J.: "Microprogramming-Another Look at Internal Computer Control", Proceedings of the IEEE, Vol.63, No.11, Nov. 1975, pp. 1554-1567.
- [77] Forgács T.; Verebély Pál: "Grafikus display és input eszközök hardware programozási kézikönyv", GD'71/T Grafikus Tervező Rendszer, MTA SzTAKI Budapest, 1974. május, 71 pp.
- [78] Forman, J.: "Gas Discharge Display as an Alternative to CRTs", Comp. Design, Vol.12, No.4. April 1973, pp. 77-83.
- [79] Foster, C.C.: "A View of Computer Architecture", Communic. of the ACM, Vol.15, No.7, July 1972, pp. 557-565.

- [80] Frank, A.J.: "B-LINE, Bell line drawing language",
Proceedings AFIPS, FJCC 1968, Vol.33, pp. 179-191.
- [81] Fraser, A.G.: "On the Interface between Computers and
Data Communications Systems", Communic. of the
ACM, Vol.15, No.7, July 1972, pp. 566-573.
- [82] Fulton, D.L.: "A Plasma-Panel Interactive Graphics
System", Proceedings of the SID, Vol. 15/2,
Second Quarter 1974, pp. 74-80.
- [83] Gallager, R.G.: "Information Theory and Reliable
Communication", John Wiley and Sons, Inc.,
New York, 1968.
- [84] Gallai, I.: "DOST-Disk Operating System for the
TPA 70/25 computers", Hung. Acad. Sci., Comp. and
Aut. Inst., Budapest, 1975, 47 pp.
- [85] Gallai I., Forgács T., Darvas P.: "TAL Assembler
nyelv TPA 70/25 számítógépre", MTA SzTAKI,
Budapest, 1974. május, 117 pp.
- [86] Gaspar, T.G. et al.: "New process language uses
English terms", Control Engineering, Vol.15,
No.10, Oct. 1968, pp. 118-121.
- [87] Gertler, J.; Sedlak, J.: "Software for Process
Control-A Survey", Automatica, Vol.11, 1975,
pp. 613-625.
- [88] Giddings, B.J.: "Contrast Enhancement with CRT and
Other Self Luminous Display Devices", IEE Conf.
on Displays, 7-10 Sept/71, Loughborough, England,
pp. 233-237.
- [89] Gladwin, B.J.: "The Utilisation of Graphic Display
Units as the Main Form of Computer Input",
The Computer Journal, No.12, 1969, pp. 114-117.
- [90] Gleichauf, P.H.; Niklas, W.F.: "CRT's for Display
Systems", Electro Technology, April 1966,
pp. 75-81.

- [91] Goede, W.F.: "Performance Differences between Cathode-Ray Tubes and Dot Matrix Displays", Proceedings of the SID, Vol.10, No.3, Summer 69, pp. 95-99.
- [92] Gooch, C.H.; Low, J.J.: "Matrix-addressed liquid crystal displays", Applied Physics, Vol.5, 1972, pp. 1218-1225.
- [93] Goodenough, J.B.: "A Lightpen-Controlled Program for on-line Data Analysis", Communic. of the ACM, Vol.8, No.2, Febr. 1965, pp. 130-134.
- [94] Goodwin, N.C.: "Cursor Positioning on an Electronic Display using Lightpen, Lightgun, or Keyboard for three basic tasks", Human Factors, 17(3), June 1975, pp. 289-295.
- [95] Goolsbey, R.A.: "APL Graphics and User Interface", Proceedings of the SID, Vol.14/2, Second Quarter 1973, pp. 73-75.
- [96] Gosch, J.: "Graphic Color Display Helps Pinpoint and Correct Industrial-Process Faults", Electronics, Vol.46, No.17, Aug. 16/73, pp. 13E.
- [97] Gouraud, H.: "Continuous Shading of Curved Surfaces", IEEE Transactions on Computers, Vol. C-20, No.6, June 1971, pp. 623-629.
- [98] Graham, S.: "Using a Standard Television Monitor as an Alphanumeric Display", Information Display, May-June/1967, pp. 59-61.
- [99] Grandbois, G.: "Improved Linear Processing Packs A-D Converter onto two IC Chips", Electronics, Vol.47, No.13, June 27/74, pp. 93-101.
- [100] Granholm, J.W.: "Alphanumeric Display Terminals", Datamation, Jan. 1976, pp. 40-52.

- [101] Gray, S.B.: "A Computer Time-Shared Display",
Information Display, Jan.-Febr./1966, pp. 50-51.
- [102] Grindley, R.E.; Dixon, C.R.; Besant, C.B.; Jebb, A.:
"Use of low cost Displays in Engineering Design",
IEE Conf. on Displays, 7-10 Sept/71, Loughborough,
England, pp. 315-318.
- [103] Grover, D.J.: "Applications of low Cost Graphic
Displays", Computer Weekly, April 9/1970, pp. 9.
- [104] Gump, J.A.: "The process picture captured: visual
aids for the operator", Inst. and Control Syst.
Vol.46, No.12, Dec. 1973, pp. 33-36.
- [105] Gurley, B.M.; Woodward, C.E.: "Light-pen Links
Computer to Operator", Electronics, Nov. 20/1959,
pp. 85-87.
- [106] Gwynn, J.W.: "CRT Terminal Access from High-Level
Languages", Proceedings of the SID, Vol.14, No.2,
2nd Quarter 1973, pp. 63-66.
- [107] Hallett, J.: "Beam-Penetration Cathode Ray Tube
Display", Proceedings of the SID, Vol.10, No.4,
Fall 69, pp. 9-14.
- [108] Hambury, J.N.; Ironside, J.; Barney, G.C.: "An Eco-
nomical Display System", The Computer Bulletin,
Vol.13, No.9, Sept. 1969, pp. 314-322.
- [109] Hanlon, A.G.: "Content-Addressable and Associative
Memory Systems, A Survey", IEEE Transaction on
Electronic Computers, Vol.EC-15, No.4, Aug. 1966,
pp. 509-521.
- [110] Haring, D.R.: "The Beam-pen: A Novel High-Speed
Input/Output Device for CRT Display Systems",
AFIPS FJCC Proceedings, Vol.27, 1965, Part I,
pp. 847-855.

- [111] Hatvany J.: "Grafikus megjelenítő eszközök",
Mérés és Automatika, XVIII.évf. 7.sz. 1970,
201-204 old.
- [112] Hatvany, J.; Newman, W.M.; Sabin, M.A.: "Report to
IIASA on a World Survey of Computer-aided Design",
July 1974, 52 pp.
- [113] Hempstead, C.F.: "Motion Perception using
Oscilloscope Display", IEEE Spectrum, Vol.3,
No.9, Sept. 1966, pp. 128-135.
- [114] Herold, E.W.: "History and Development of the Color
Picture Tube", Proceedings of the SID, Vol.15,
No.4, Fourth Quarter 1974, pp. 141-149.
- [115] Hirsch, R.S.: "A Choice of Displays in an Information
Retrieval System", IEE Conf. on Displays, 7-10
Sept/71, Loughborough, England, pp. 283-288.
- [116] Hlady, A.M.: "A disc-based multiple terminal display
system", SID Journal, Vol.10, No.6,
Nov.-Dec./1973, pp. 9-12,22.
- [117] Hobbs, L.C.: "Present and Future State-of-the-Art in
Computer Memories", IEEE Transactions on
Electronic Computers, Vol.15, Aug. 1966,
pp. 534-550.
- [118] Hoffman, R.L.; Skuldt, E.L.: "Superposition of
Multiple Independent Matrix-Display Symbol Sets",
IBM Technical Disclosure Bulletin, Dec. 1971,
Vol.14, No.7, pp. 2000-2001.
- [119] Hofstein, S.R.; Rudisill, W.A.: "Silicon-Target
Storage Tubes Outdo Direct-View Types in
Versatility", Electronics, Vol.46, No.4,
Febr.15/73, pp. 91-94.
- [120] Honig, P.: "Principles of Sugar Technology", Elsevier
Publishing Co., Vol. I, 1953.

- [121] Honig, P.: "Principles of Sugar Technology", Elsevier Publishing Co., Vol.II, 1959.
- [122] Honig, P.: "Principles of Sugar Technology", Elsevier Publishing Co., Vol.III, 1963.
- [123] Horváth, I.; Nagy, M.: "TPA-70 Minicomputer Software, Assembler User's Manual", Hung. Acad. Sci, Central Research Inst. for Physics, 1974, 147 pp.
- [124] Huddleston, H.F.: "An Evaluation of Alphanumerics for a 5x7 Matrix Display", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 145-147.
- [125] Hughes, A.D.: "Desired Characteristics of Automated Display Consoles", Proceedings of the SID, Vol.10, No.1, Winter 1969, pp. 46-50.
- [126] Jervis, M.W.: "CRT Displays of Graphs in CEGB Power Stations", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 55-62.
- [127] Jones, I.A.: "Display Terminal Design", The Computer Bulletin, Vol.13, No.11, Nov. 1969, pp. 391-395.
- [128] Jordan, B.W.; Barrett, R.C.: "A Cell Organized Raster Display for Line Drawings", Communic. of the ACM, Febr.1974, Vol.17, No.2, pp. 70-77.
- [129] Jordan, B.W.; Barrett, R.C.: "A Scan Conversion Algorithm with Reduced Storage Requirements", Communic. of the ACM, Vol.16, No.11, Nov. 1973, pp. 676-682.
- [130] Karp, H.R.: "Digital-to-Analog Converters: Trading-off bits and bucks", Electronics, Vol.45, No.6, March 13/1972, pp. 84-90.
- [131] Kashman, M.: "Display Buyer's Guide", Control Eng., June 1968, pp. 111-114.

- [132] Kawaji, A.; Ando, T.; Shiraki, H.; Okubo, T.:
"A Solid-State Matrix Display Employing Gallium Phosphide Diodes", Proc. of the SID 7th National Symposium, 1966, Los Angeles, pp. 255-262.
- [133] Kaye, D.N.: "Display method draws curves, not vectors, and with less data", Electronic Design 22, Oct. 25/1973, pp. 32,34.
- [134] Keast, D.N.: "A Survey of Graphic Input Devices for CAD", Machine Design, Aug. 3/1967, pp. 114-120.
- [135] Kesselman, M.: "An Improved Light Gun Tracking Algorithm Based on a Recursive Digital Filter", Proceedings of the SID, Vol.14, No.2, 2nd Quarter 1973, pp. 52-61.
- [136] Kinney, G.C.: "Symbols, Sense, and Sin", Proceedings of the SID, Vol.10, No.1, Winter 1969, pp. 30-33.
- [137] Kinney, G.C.; Showman, D.J.: "The relative legibility of uppercase and lowercase typewritten words", Information Display, Sept.-Oct./1967, pp. 34-39.
- [138] Knowlton, K.: "A Report on the use of Fortran-Coded EXPLOR for the Teaching of Computer Graphics and Computer Art", Proc. of Symp. on two-dimensional man-machine communication, ACM SIGPLAN Notices, Vol. 7/10, Oct. 1972, pp. 103-112.
- [139] Koehler, H.F.: "Advances in Memory Technology", Computer Design, Vol.13, No.6, June 74, pp. 71-77.
- [140] Konkle, K.H.: "An Analog Comparator as a Pseudo-Light Pen for Computer Displays", IEEE Trans. on Computers, Vol.C-17, No.1, Jan. 1968, pp. 54-55.
- [141] Kossiakoff, A.; Sleight, T.P.: "A Programming language for real-time systems", Fall Joint Computer Conf., 1972, pp. 923-942.

- [142] Kulsrud, H.E.: "A general purpose graphic language",
Communic. of the ACM, Vol.11, No.4, April 1968,
pp. 247-254.
- [143] Kutsuzawa, J.; Saida, M.; Ando, T.: "Color Graphic
Display with Surface Painting", FUJITSU Scientific
and Technical Journal, June 1975, pp. 1-14.
- [144] Lábadi, A.: "The GDIO-Graphical Display Input/Output
System, Programming Manual", Hung. Acad. Sci.,
Computer and Automation Institute, Nov. 1974.
48 pp.
- [145] Lábadi A.: "GD'71T Felhasználói Ismertető, Grafikus
Display Input-Output Rendszer", MTA SzTAKI,
Budapest, 1976. március, 3. kiadás, 49 old.
- [146] Lamoureux, W.R.: "A New Approach to Character Genera-
tion", 9th National Symposium on Information
Display, May 1968, pp. 55-60.
- [147] Larach, S.; Hardy, A.E.: "Cathode-Ray-Tube Phosphors:
Principles and Applications", Proc. of the IEEE,
Vol.61, No.7, July 1973, pp. 915-926.
- [148] Lidinsky, W.P.; Becker, J.A.: "A Computer-Driven
Full-Color Raster scan Display System",
IEEE Transactions on Broadcasting and TV
Receivers, Vol.BTR-18, No.1, 1972, pp. 26-31.
- [149] Little, J.L.: "ASCII Code Applications to Alpha-
numeric Display Terminals", Proc. of the SID,
Vol.13/3, Third Quarter 1972. pp. 154-160.
- [150] Little, J.L.: "ASCII Code Applications to Alpha-
numeric Display Terminals", Proc. of the SID,
Vol. 14/2, 2nd Quarter 1973, pp. 67-72.
- [151] Lloyd, G.R.; Van Dam, A.: "Design Considerations for
Microprogramming Languages", AFIPS, National
Computer Conf. 1974, Proceedings, Vol.43, 1974.
pp. 537-542.

- [152] Locascio, J.T.; Karanza, G.L.; Dalton, J.J.:
"Obtaining Light Pen Versatility", Information
Display, Nov-Dec./1967, pp. 36-39.
- [153] Loewe, R.T.; Sisson, R.L.; Horowitz, P.: "Computer
Generated Displays", Proc. of the IRE, Jan. 1961,
pp. 185-195.
- [154] Lovercheck, L.R.: "Raster-scan technique provides
multicolor graphic displays", Electronics,
Vol.45, No.12, June 5/1972, pp. 111-116.
- [155] Luhowy, G.M.: "Characteristics of Computer Driven
CRT Displays", Telecommunications, Febr. 1971,
pp. 27, 52.
- [156] Lukaszewicz, L.: "EOL-A symbol manipulation
language", The Computer Journal, Vol.10,
1967-1968, pp. 53-59.
- [157] Luxemburg, H.R.; Bonness, Q.L.: "Quantitative
Measures of Display Characteristics", Information
Display, July - Aug./1965, pp. 8-12.
- [158] Macaulay, M.: "Low-Cost Terminals Using Television
Techniques", National Radio and Electronics Eng.
Convention-Abstracts, IREE Australia, May 1967,
pp. 222.
- [159] Macaulay, M.: "A Low Cost Computer Graphic Terminal",
Proceedings 1968, Fall Joint Computer Conf.
AFIPS, Vol.33, Dec. 1968, pp. 777-785.
- [160] Mac Dougall, M.H.; McAlpine, J.S.: "Computer System
Simulation with ASPOL", SIGM-ACM, Symposium on
the Simulation of Computer Systems, Gaithersburg
June 1973, pp. 93-103.
- [161] Machida, H.; Fuse, Y.: "The Gain in the Definition
of Color CRT Image Display by the Aperture-
Grille", IEEE 1972, Conf. on Display Devices,
pp. 101-108.

- [162] Manna, Z.; Waldinger, R.J.: "Toward Automatic Program Synthesis", Communications of the ACM, Vol.14, No.3, March 1971, pp. 151-165.
- [163] Marcus, A.: "A prototype Computerized Page-design System", Visible Language, V3 Summer 1971, pp. 197-210.
- [164] Marshall, W.; Brown, C.: "Take a bit of advice: use 16-bit converters carefully", Electronics, Vol.45, No.21, Oct. 9/1972, pp. 96-99.
- [165] Marti, J.B.: "ELMOL: A Language for the Real Time Generation of Electronic Music", SIGPLAN Notices, July 1973, pp. 23-30.
- [166] Martin, A.F.: "Penetration Color tubes are enhancing information displays", Electronics, Vol.46, No.2, Jan. 18/1973, pp. 155-160.
- [167] Martin, L.: "Use your Oscilloscope for Numeric Display", Electronic Design 26, Dec. 23/1971, pp. 64-65.
- [168] Martin, M.J.: "A General Purpose Plotting Language and Interpreter Program for Remote Plotting On-line to George 3", Royal Aircraft Establishment, Technical Report No.74109, Nov. 1974, 27 pp.
- [169] Martin, T.J.; Sykes, A.: "Interactive Computing Helps Study Problems", Computer Weekly, Aug. 10/1972, pp. 6-7.
- [170] Martino, F.J.; Manne, R.E.: "Interactive Color CRT terminals take on more supervisory duties", Can. Controls and Instrum. Vol.14, No.4, April 1975, pp. 22-25.
- [171] Mattox, J.: "External gate doubles counter speed", Electronics, Vol.47, No.3, Feb. 7/1974, pp. 102.

- [172] McCrea, P.G.; Maxwell, P.C.: "Low cost video display systems", 6th Australian Computer Conf. Vol.I, Sydney, Australia, 20-24 May/1974, pp. 108-118.
- [173] McGregor, P.: "Effective use of Data Communications Hardware", AFIPS, National Computer Conf. 1974, Proceedings, Vol.43, pp. 565-575.
- [174] Meahl, E.M.: "Counter Look-Ahead Techniques-Part 1: Stretch Counter Size", Electronic Design 25, Dec. 6/73, pp. 104-108.
- [175] Meahl, E.M.: "Counter Look-Ahead Techniques-Part 2: Expand Counter Capability", Electronic Design 26, Dec. 20/73, pp. 68-75.
- [176] Merritt, M.J.; Miller, D.S.: "MOBSSL-UAF-An augmented block structural continuous system simulation language for digital and hybrid computers", Fall Joint Comp. Conf. 1969, pp. 255-274.
- [177] Metzger, R.A.: "Computer generated graphic segments in a raster display", Proc. AFIPS, 1969 Spring Joint Comp. Conf. Vol. 34, pp. 161-172.
- [178] Michener, J.C.; Van Dam, A.: "Storage Tube Graphics: A Comparison of Terminals", Int. Symp. on Computer Graphics 70, Vol.3, April 1970, pp. 851-886.
- [179] Miller, J.C.; Wine, C.M.: "A Simple Display for Characters and Graphics", IEEE Transactions on Computers, Vol.C-17, No.5, May 1968, pp. 470-475.
- [180] Miller, J.C.; Wine, C.M.: "A Light Pen for Remote Timeshared Graphic Consoles", IEEE Transactions on Computers, Vol.C-17, No.8, Aug. 1968, pp. 799-802.
- [181] Miller, S.W.: "Display Requirements for Future Man-Machine Systems", IEEE Transactions on Electron Devices, Vol.ED-18, No.9, Sept. 1971, pp.616-620.

- [182] Miller, W.E.: "Digital Computer Applications to Process Control", Proc. of the 1st Int. Conf. IFAC-IFIP, Sweden, 1965, pp. 55-71.
- [183] Miller, W.F.; Shaw, A.C.: "Linguistic Methods in Picture Processing, A Survey", Proc. AFIPS, FJCC, 1968, Vol.33, pp. 279-290.
- [184] Millichamp, D.: "Dress up displays in color for greater attraction", Engineer /GB/, Vol.235, No.6092, Dec. 14/1972. pp. 34-35,37.
- [185] Mohan Rao, K.C.: "The Use of Computer Driven Displays in the Supervision and Control of Large Power Systems", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 277-282.
- [186] Morpurgo, R.; Sami, M.: "Some Solutions to the Problems of Defining and Compiling Graphic Languages", Revue Francaise d'Informatique et de Recherche Operationnelle, Num. B-1, pp. 21-30.
- [187] Mrazek, D.; Morris, M.: "PLAs replace ROMs for logic designs", Electronic Design 22, Oct. 25/1973, pp. 66-70.
- [188] Muoio A.W.: "Character/Symbol Generation by MOS Read Only Memory", Proceedings of the SID, Vol.11/1, First Quarter 1970, pp. 6-15.
- [189] Mumma, F.W.: "Auerbach on Computer Technology: Progress with Displays", Data Processing Magazine, Oct. 1969, pp. 34.
- [190] Myer, T.H.; Sutherland, I.E.: "On the Design of Display Processors", Communic. of the ACM, Vol.11, No.6, June 1968, pp. 410-414.
- [191] Narasimham, P.V.H.M.L.: "Varying Beam Thickness in CRT Display Systems", Electronics, Vol.46, No.21, Oct. 11/73, pp. 123.

- [192] Newey, M.C.; Poole, P.C.; Waite, W.M.: "Abstract Machine Modelling to Produce Portable Software-A Review and Evaluation", Software-Practice and Experience, Vol.2, John Wiley and Sons, Ltd. 1972, pp. 107-136.
- [193] Newman, W.M.: "Display Procedures", Communic. of the ACM, Vol.14, No.10, Oct. 1971, pp. 651-660.
- [194] Newman, W.M.: "Raster-scan graphics in CAD" Xerox Palo Alto Research Center, Oct. 1975, 7 pp.
- [195] Newman, W.M.; Sproull, R.F.: "An Approach to Graphic System Design", Proceedings of the IEEE, Vol.62, No.4, April 1974, pp. 471-483.
- [196] Nezu, K.; Naito, S.: "An Effective System for Displaying a Large Character Set", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 249-256.
- [197] Nickerson, R.S.: "Man-Computer Interaction: "A Challenge for Human Factors Research", Ergonomics, Vol.12, No.4, 1969, pp. 501-517.
- [198] Noll, A.M.: "Scanned-Display Computer Graphics", Communic. of the ACM, Vol.14, No.3, March 1971, pp. 143-150.
- [199] Notley, M.G.: "A Graphical Picture Drawing Language", The Computer Bulletin, March 1970, pp. 68-74.
- [200] Obádovics J.G.: "Matematika", Műszaki Könyvkiadó, Budapest, 1962.
- [201] Oetzmann, E.H.; Radley, D.E.: "A Modular Interactive Display System for use as a Local or Remote Terminal", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 13-17.

- [202] Ogden, S.; Wadsworth, N.: "On-line Graphics at the University of Utah", Datamation, Nov. 1969, pp. 159-165.
- [203] Ogden, S.: "The Evans and Sutherland LDS-1 Display System", Proceedings of the SID, Vol.11/2, 2nd Quarter 1970, pp. 52-60.
- [204] Page, P.B.: "The Effects of Electronic Circuits on Alphanumeric Display Capabilities and Costs", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 97-101.
- [205] Papp G.: "Nagy Erőművi Blokkok Számítógépes Irányítása", MTA KFKI, RTF 5/76, Budapest, 1976. márc., 36 old.
- [206] Parker, D.B.: "Solving Design Problems in Graphical Dialogue", On-line Computing, Ed. by Walter J. Karplus, McGraw Hill, 1967, pp. 179-219.
- [207] Parslow, R.D.; Prowse, R.W.; Elliot Green, R.: "Computer Graphics, Techniques and Applications", Plenum Press, London, 1969.
- [208] Powley, C.: "TV could replace control room mimic diagrams", Process Eng. Plant and Contr. Dec. 1971, pp. 43.
- [209] Ramamoorthy, C.V.; Kim, K.H.: "Pipelining - the Generalized Concept and Sequencing Strategies", AFIPS, National Computer Conf. 1974, Proceedings, Vol.43, pp. 289-297.
- [210] Raphael, H.A.: "Fast BCD/binary conversions", Electronic Design 22, Oct. 25/1973, pp. 84-89.
- [211] Reetz, J.: "Communication man-process-computer. Problems and solutions for dialogue and control centre planning", Interkama 5th Int. Congress with Exhibition for Instruments and Automation. Dusseldorf, Germany, 14-21 Oct./1971, pp. 198-207.

- [212] Reingold, I.: "Display Devices: A Perspective on Status and Availability", Proc. of the SID, Vol.15/2, Second Quarter 1974, pp. 63-73.
- [213] Rényi I.; Törő F.; Vajda F.: "Fényceruza: Az ember-gép kapcsolat hatékony eszköze", Mérés és Automatika, XVIII. évf. 2.sz. 1970, 47-52 old.
- [214] Rényi I.; Törő F.; Vajda F.: "Kisszámítógépvezérelt rászter-rendszerű adatmegjelenítő berendezés", Mérés és Automatika XVIII. évf. 7.sz. 1970. 208-213 old.
- [215] Resch, R.D.: "The topological design of sculptural and architectural systems", Nat. Comp. Conf. and Exposition, AFIPS, Conf. Proc., Vol.42. June 1973, pp. 643-650.
- [216] Reyling, G.: "PLAs Enhance Digital Processor Speed and Cut Component Count", Electronics, Vol.47, No.16, August 8/1974, pp. 109-114.
- [217] Rheinboldt, W.C.; Basili, V.R.; Mesztényi, C.K.: "On a Programming Language for graph Algorithms", Computer Science Center, Univ. of Maryland, 1972, pp. 221-241.
- [218] Riezenman, M.J.: "Special Report: The new displays complement the old", Electronics, Vol.46, No.8, April 12/1973, pp. 91-99.
- [219] Riggs, A.L.; Yows, P.W.: "The design of a multi-purpose display console for command and control systems", 1972 SID Symp. Digest of Technical Papers, San Francisco, California, 6-8 June/1972, pp. 26-27.
- [220] Riley, W.B.: "Special-Report: Semiconductor memories are taking over data-storage applications", Electronics, Vol.46, No.16, Aug. 2/1973, pp. 75-90.

- [221] Riley, W.B.: "Special Report: The Technology gap starts to close for computer peripherals", Electronics, Vol.45, No.16, July 31/1972, pp. 59-74.
- [222] Roberge, J.K.; King, P.A.: "An Economical Approach to High-Speed Character Generation and Display", The 1970 IDEA Symposium, Digest of Papers, pp. 104-105.
- [223] Roberts, L.G.: "Graphical Communication and Control Languages", 2nd Congress on the Information System Sciences, Spartan Books, Inc., MacMillan and Co., Wash.London, 1965, pp. 211-217.
- [224] Rose, G.A.: "Computer Graphics Communication Systems", Proc. of IFIP Congress 1968, Vol.2, Aug. 1968, pp. 692-703.
- [225] Rosenzweig, W.: "An eleven-switch Decimal-to-Seven Segment Decoder", Proc. of the SID, Vol.13, No.4, 4th Quarter 1972, pp. 185-187.
- [226] Rutland, D.: "A Digital Interactive Color Television Display", Information Display, Sept.-Oct./1970, pp. 20-22,31.
- [227] Salton, G.: "Dynamic Document Processing", Communic. of the ACM, Vol.15, No.7, July 1972, pp. 658-668.
- [228] Sammet, J.E.: "Programming Languages: History and Future", Communic. of the ACM, Vol.15, No.7, July 1972, pp. 601-610.
- [229] Sammet, J.E.: "Programming Languages: History and Fundamentals", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1969, pp. 20-22,726.
- [230] Sammet, J.E.: "Problems in, and a pragmatic approach to, programming language measurement", Fall Joint Comp. Conf. 1971, pp. 243-251.

- [231] Sammet, J.E.: "An overview of programming languages for specialized application areas", AFIPS, Spring Joint Computer Conf. Proc., Vol.40, May 1972, pp. 299-311.
- [232] Sammet, J.E.: "Roster of Programming Languages 1968", Computer and Automation, Vol.17, No.6, June 1968, pp. 120-123.
- [233] Sammet, J.E.: "Roster of Programming Languages 1969", Computers and Automation, 1969 Directory, pp. 153-158.
- [234] Sammet, J.E.: "Roster of Programming Languages 1970", Computers and Automation 19, 6B, Nov. 1970. pp. 6-11,21.
- [235] Sammet, J.E.: "Roster of Programming Languages 1971", Computers and Automation, Vol.20,6B, June 1971, pp. 6-13.
- [236] Sammet, J.E.: "Roster of Programming Languages 1972", Computers and Automation 1972 Directory, Vol.21, Aug. 1972, 6B, pp. 123-132.
- [237] Sammet, J.E.: "Roster of Programming Languages for 1973", Computer Reviews, April 1974, pp. 147-160.
- [238] Saxon, R.: "Plasma's Progress: Gas-Discharge Technology Moves into Analog Realm", Electronics, Vol.47, No.5, March 7/74, pp. 89-93.
- [239] Schellstede, G.: "Man-computer communication by displays and keyboards for load dispatching-operation", Elektrotech. Z. Vol.93, No.5, May 1972, pp. 298-301.
- [240] Schindler, A.: "Computer CAMAC Based Control System for a 1500 l/s Waterpump Station", Proc. of the 1st International Symp. on CAMAC, Issue 9, April 1974, pp. 227-232.

- [241] Schoeffler, J.D.: "Process Control Software", Datamation, Vol.12, No.2, Febr. 1966, pp. 33-42.
- [242] Schoeffler, J.D.; Willmott, T.L.; De Dourek, J.: "Programming Languages for Industrial Process Control", Proc. IFAC, Symp. on Computer Control, June 1967.
- [243] Schoeffler, J.D.; Temple, R.H.: "A Real-Time Language for Industrial Process Control", Proc. of the IEEE, Vol.58, No.1, Jan. 1970, pp. 98-111.
- [244] Schultz, G.W.; Holt, R.M.; McFarland, H.L.: "A Guide to using LSI Microprocessors", Computer, June 1973, pp. 13-19.
- [245] Schwinn, P.M.: "A problem oriented graphic language", Proc. of 22nd Nat. Conf. ACM, 1967, pp. 471-477.
- [246] Scrupski, S.E.: "A Microcomputer for Industrial use", Electronics, Vol.49, No.10, May 13/1976, pp. 117.
- [247] Seats, P.: "The Cathode Ray Tube - A Review of Current Technology and Future Trends", IEEE 1970 Conf. on Display Devices, pp. 116-120.
- [248] Segallis, B.: "Special Report on Memories", Electronic Products, Jan. 1968. pp. 19-26.
- [249] Sheffield, H.E.: "Color CRT Terminals for Supervisory Control", Computer Design, March 1972, pp. 91-96.
- [250] Sherr, S.: "Fundamentals of Display System Design", Wiley-Interscience, John Wiley and Sons. Inc., New York, 1970.
- [251] Sherr, S.: "Digital Television - A Low Cost Approach to Multiterminal Graphics", Computer Graphics 70, Vol.11, April 1970, pp. 49-67.
- [252] Sherr, S.: "The Technology and Characteristics of Computer Driven, Interactive Graphic Display Systems", Proc. of the SID, Vol.15/1, 1st Quarter, pp. 2-16.

- [253] Sherr, S.: "CRT Displays: Types and Capabilities",
SID, Int. Symp. Digest of Technical Papers 1974,
Vol.5, pp. 174-176.
- [254] Sherr, S.: "Applications of Digital Television
Displays to Command and Control", Proc. of the
SID, Vol.11/2, 2nd Quarter 1970, pp. 61-70.
- [255] Sherr, S.: "Computer Graphics Technology and
Available Equipment", Lecture Notes for Geo-
graphical Data Based Systems, Seminar, 1973.
- [256] Shopiro, J.E.: "Survey of Computer Graphics",
Technical Report No.1, Appendix A., The Produc-
tion Automation Project, Univ. of Rochester
Jan. 1974, 50 pp.
- [257] Shurtleff, D.A.: "Legibility Research", Proc. of the
SID, Vol.15/2, Second Quarter 1974, pp. 41-52.
- [258] Shurtleff, D.A.: "Studies in Television Legibility -
A Review of the Literature", Information Display,
Vol.4, No.1, Jan.-Feb./1967.
- [259] Sobel, A.: "Some Constraints on the Operation of
Matrix Displays", IEEE Transactions on Electron
Devices, Sept. 1971, pp. 797-798.
- [260] Sobel, A.: "Selection Limits in Matrix Displays",
IEEE Conf. Record of 1970 IEEE Conf. on Display
Devices, pp. 74-84.
- [261] Sparks, J.E.: "Television that Nobody Watches",
Machine Design, Febr.10/72, Vol.44, pp. 93-97.
- [262] Spitzer, J.F.; Robertson, J.G.; Neuse, D.H.: "The
Colingo System Design Philosophy", 2nd Congress
on the Information System Sciences, pp. 33-47.
- [263] Stevens, W.W.: "Magnetic disc + TV monitors = low
cost graphic display terminal", Inf. Display,
Vol.5, No.1, Jan.-Feb./1968, pp. 45.

- [264] Stotz, R.H.; Cheek, T.B.: "A Low-Cost Graphic Display for a Computer Time-Sharing Console", 8th Nat. Symp. on Inf. Display, May 1967, pp. 91-100.
- [265] Stupar, T.: "Characterization of light pen sensitivity", Inf. Display, May-June/1967, pp. 67-69.
- [266] Summerhill, S.: "Black-white display terminals adapted for color by controller", Electronic Design 22, Oct. 25/1973, pp. 112.
- [267] Sutherland, I.E.: "Future Trends in Computer-Aided Design", Proc. of the SID, Vol.11/2, 2nd Quarter 1970, pp. 49-51.
- [268] Sutherland, W.R.: "The CORAL language and Data Structure in on-line specification of computer procedures", M.I.I. Lincoln Lab., Techn. Report No.405, 1966.
- [269] Szabó P.: "A TPA 70 Kiszámítógép párhuzamos-soros távgépiró kódátalakítója", MTA KFKI-74-74, Budapest, 1973, október, 9 old.
- [270] Talbot, P.A.; Carr, J.W.; Coulter, R.R.; Hwang, R.C.: "Animator: An On-line two-dimensional Film Animation System", Communic. of the ACM, Vol.14, No.4, April 1971, pp. 251-259.
- [271] Taylor, F.E.: "Computer-Driven Displays and the Display Group", The Computer Bulletin, Vol.15, No.1, Jan. 1971, pp. 4-8.
- [272] Teichroew, D.; Sayani, H.: "Automation of System Building", Datamation, Aug. 15/1971, pp. 25-30.
- [273] Thiel, R.; Heinrich, G.: "Supervision and Control in Utility Centres", Proc. of the 1st International Symp. on CAMAC, Issue 9, April 1974, pp. 261-267.
- [274] Thomas, E.M.: "Display Programming-1968", Proc. of the SID, Vol.10, No.1, Winter/69, pp. 34-42.

- [275] Thompson, D.A.: "Man-computer system: toward balanced co-operation in intellectual activities", Tech. Report, Stanford Univ. 1968, 12 pp.
- [276] Thompson, F.B.; Dostert, B.H.: "The future of specialized languages", AFIPS Spring Joint Computer Conf. Vol.40, 1972, pp. 313-319.
- [277] Thornhill, D.E.; Cheek, T.B.: "Raster-scan tube adds to flexibility and lower cost of graphic terminal", Electronics, Vol.47, No.3, Febr.7/1974, pp. 95-101.
- [278] Ueberschar, G.: "The operating system for the Siemens 300 process computers", Siemens Rev. Vol.39, No.6, June 1972, pp. 261-264.
- [279] Van Dam, A.: "Computer Driven Displays and their use in Man/Machine Interaction", Advances in Computers, Vol.7, 1966, pp. 239-290.
- [280] Van Dam, A.; Evans, D.: "A compact data structure for storing, retrieving and manipulating line drawings", AFIPs, Spring Joint Comp. Conf., April 1967, Vol.30, pp. 601-610.
- [281] Vartebedian, A.G.: "Effects of Parameters of Symbol Formation on Legibility", Information Display, May 1970, pp. 23-26.
- [282] Walker, G.M.: "Color Television Picture Tubes", Electronics, Vol.47, No.16, August 8/1974, pp. 120-122.
- [283] Wasmund, S.: "The colour television monitor as an interactive computer peripheral", Elektron. Anz., Vol.7, No.3, March 1975, pp. 67-69.
- [284] Weisberg, D.E.: "Man-machine communication and Process Control", Data Proc. Magazine, Sept. 1967, pp. 18-24.

- [285] Weissberger, A.J.: "Microprocessors in the processing plant", IEEE Trans. Ind. Electron. and Control Instrum. Vol. IECI-22, No.3, Aug. 1975.
pp. 354-358.
- [286] Wightman, E.J.: "Developments in Computer Control Systems for Machine Tools", Proc. Inst. Mech. Eng. 1974, Vol.188, 6/74, pp. 49-75.
- [287] Williams, C.M.: "Horizontal vs. Vertical Display of Numbers", Human Factors, Vol.8, June 1966,
pp. 237-238.
- [288] Williams, D.C.: "Graphic Display Facilities for on-line Process Control", IEE Conf. on Displays, 7-10 Sept/71, Loughborough, England, pp. 319-323.
- [289] Williams, R.: "A Survey of Data Structures for computer graphics systems", Computing Surveys, Vol.3, No.1, March 1971, pp. 1-21.
- [290] Williams, R.; Krammer G.: "EX-GRAF: An Extensible Language Including Graphical Operations", Computer Graphics and Image Processing, 1972, 1, pp. 317-340.
- [291] Wolf, J.W.; Williams, J.H.: "On the beam for sharp crt character displays", Electronics, Vol.42, No.20, Sept. 29/1969, pp. 108-111.
- [292] Woods, M.A.: "The Light Pen: A graphic input device for computer driven displays", IEE Conference Publications No.32, 1967, pp. 230-237.
- [293] Yamazaki, E.; Ohkawa, K.: "Color Graphical Display System", SID Journal, May-June/1974,
pp. 14-15, 18-21.
- [294] Zarándi L.: "Nagy Erőművi Blokkok Számítógépes Irányítása", MTA KFKI RTF 7/76, Budapest, 1976. március, 45 old.

- [295] Zimmermann, R.: "Grafische Darstellungen bei Datensichtgeräten mit Anzeige nach dem Fernsehprinzip", Regelungstechnische Praxis 1975, Heft 6, pp. 167-176.
- [296] Zimmermann, R.; Etschberger, K.: "Optical display system for process supervision", Regelungstechn. Prax. and Prozess-Rechentech. Vol.16, No.8, Aug. 1974, pp. 203-209.
- [297] "Color Graphics System Combines Char-Oriented and Graphics Capabilities", Computer Design, July/74, Vol.13, No.7, pp. 20.
- [298] "Color TV tube has black stripes for sharp picture", Electronics, Vol.45, No.24, Dec. 4/1972, pp. 3E.
- [299] "Computer Control Becoming Increasingly Involved in Plastics Processing Industry", Computer Design, Jan. 1972, pp. 30,32. Digital Control Systems, Design and Applications.
- [300] Instruments and Control System Staff: "Digital-to-Analog Converter Survey", Instruments and Control System, Nov. 1971, pp. 89-95.
- [301] "Minicomputer Provides Broad Process Control Capabilities", Computer Design, May 1972, pp. 50. Digital Control Systems, Design and Applications.
- [302] "Minicomputer Control Gas Pipeline Flow", Computer Design, May 1972, pp. 44,46. Digital Control Systems, Design and Applications.
- [303] "Process-Control Display Replaces Tote Boards, Meters", Electronics, New Products, Vol.47. No.11, May 30/74, pp. 140.
- [304] "Special Issue: The Great Takeover", Electronics, Vol.46, No.22, Oct. 25/1973, pp. 69-206.

- [305] "Updating Circuit Symbols, the Graphic Language of Electronics", Electronics, Vol.48, No.7, April 3/75, pp. 90-98.
- [306] TPA-70 Minicomputer System, 70-1025 Central Processor Reference Manual, Hung. Acad. Sci., Central Res. Inst. for Physics, 1974.
- [307] Marketing Brochure of the Aydin Controls Color CRT Display Systems, USA, 5000 Series Raster-Scan Displays.
- [308] CG1000 Seven Colour Graphic Display Peripheral, Marketing Brochure 9-407, CIT ALCATEL, Groupe CGE, France.
- [309] VT30 Colour Mimic Diagram Display System for the PDP-11, Digital Equipment Corp., USA.
- [310] Hudson, R.G.: "The Engineers Manual", John Wiley and Sons, Inc., 1961, 340 pp.
- [311] Lehmann, C.H.: "Geometría Analítica", Edición Revolucionaria, Inst. del Libro, 1965, 494 pp.

Appendix 1

Example of a Pump Station

Virgin Program

1	0002		S TANK
2	0002	01FD	M (0,-1)
3	0004	100D	T (+V,B,1) T
	0006	D400	
4	0008	24F7	E (+H,R,2) HL
	000A	0200	
5	000C	1421	E (-V,R,1) 2 ROMB
	000E	0100	
6	0010	24F1	E (-V,R,2) VL
	0012	0300	
7	0014	1433	E (-H,R,1) 3 ROMB
	0016	0100	
8	0018	2453	E (-H,R,2) 5 CROS
	001A	0000	
9	001C	1405	E (+V,R,1) 0 ROMB
	001E	0100	
10	0020	24F5	E (+V,R,2) VL
	0022	0300	
11	0024	1417	E (+H,R,1) 1 ROMB
	0026	0100	
12	0028	0100	EOS
13	002A		S HV
14	002A	34F3	E (-H,R,3) HL
	002C	0200	
15	002E	14F3	E (-H,R,1) DIOD3
	0030	0700	
16	0032	14F3	E (-H,R,1) DIOD1
	0034	0500	
17	0036	34F3	E (-H,R,3) HL
	0038	0200	
18	003A	07FD	M (3,-1)
19	003C	100F	T (+H,B,1) V
	003E	D600	
20	0040	0100	EOS
21	0042		S VV
22	0042	14F5	E (+V,R,1) VL
	0044	0300	
23	0046	14F5	E (+V,R,1) DIOD4
	0048	0800	
24	004A	14F5	E (+V,R,1) DIOD2
	004C	0600	
25	004E	14F7	E (+H,R,1) VL
	0050	0300	
26	0052	100F	T (+H,B,1) V
	0054	D600	
27	0056	0100	EOS
28	0058		S PUMP2
29	0058	0002	C TANK
	005A	0002	
30	005C	1471	E (-V,R,1) 7 CROS
	005E	0000	
31	0060	1009	T (-V,B,1) P
	0062	D000	
32	0064	01FD	M (0,-1)
33	0066	14F7	E (+H,R,1) HL
	0068	0200	
34	006A	1427	E (+H,R,1) 2 CROS
	006C	0000	
35	006E	14F1	E (-V,R,1) VL
	0070	0300	
36	0072	1463	E (-H,R,1) 6 CROS
	0074	0000	
37	0076	1403	E (-H,R,1) 0 CROS
	0078	0000	

38	007A	05FD	M (2,-1)
39	007C	0100	E08
40	007E		S F1
41	007E	34F3	E (-H,R,3) HL
	0080	0200	
42	0082	1411	E (-V,R,1) 1 CROS
	0084	0000	
43	0086	0100	E08
44	0088	3A0A	P 29,5
45	008A	14F3	E (-H,R,1) ROMB
	008C	0100	
46	008E	74F3	E (-H,R,7) HL
	0090	0200	
47	0092	74F3	E (-H,R,7) HL
	0094	0200	
48	0096	1405	E (+V,R,1) 0 CROS
	0098	0000	
49	009A	14F5	E (+V,R,1) VL
	009C	0300	
50	009E	1901	M (12,0)
51	00A0	1433	E (-H,R,1) 3 CROS
	00A2	0000	
52	00A4	74F3	E (-H,R,7) HL
	00A6	0200	
53	00A8	84F3	E (-H,R,8) HL
	00AA	0200	
54	00AC	1405	E (+V,R,1) 0 CROS
	00AE	0000	
55	00B0	0002	C VV
	00B2	0042	
56	00B4	05F5	M (2,-3)
57	00B6	0002	C VV
	00B8	0042	
58	00BA	05F5	M (2,-3)
59	00BC	0002	C VV
	00BE	0042	
60	00C0	05F5	M (2,-3)
61	00C2	0002	C VV
	00C4	0042	
62	00C6	75F1	M (-6,-4)
63	00C8	1477	E (+H,R,1) 7 CROS
	00CA	0000	
64	00CC	07FD	M (3,-1)
65	00CE /	14F1	E (-V,R,1) VL
	00D0	0300	
66	00D2	1477	E (+H,R,1) 7 CROS
	00D4	0000	
67	00D6	070D	M (3,3)
68	00D8	54F5	E (+V,R,5) VL
	00DA	0300	
69	00DC	1EF5	E (+V,Y,1) DIOD4
	00DE	0800	
70	00E0	64F5	E (+V,R,6) VL
	00E2	0300	
71	00E4	1423	E (-H,R,1) 2 CROS
	00E6	0000	
72	00E8	A4F3	E (-H,R,10) HL
	00EA	0200	
73	00EC	54F3	E (-H,R,5) HL
	00EE	0200	
74	00F0	1EF3	E (-H,Y,1) DIOD1
	00F2	0500	
75	00F4	0002	C HV
	00F6	002A	

76	00F8	7905	M (-4,1)
77	00FA	14F5	E (+V,R,1) ROMB
	00FC	0100	
78	00FE	1701	M (11,0)
79	0100	0002	C VV
	0102	0042	
80	0104	05F5	M (2,-3)
81	0106	0002	C VV
	0108	0042	
82	010A	7DF1	M (-2,-4)
83	010C	1471	E (-V,R,1) 7 CROS
	010E	0000	
84	0110	7901	M (-4,0)
85	0112	14F1	E (-V,R,1) VL
	0114	0300	
86	0116	1447	E (+H,R,1) 4 CROS
	0118	0000	
87	011A	74F7	E (+H,R,7) HL
	011C	0200	
88	011E	1421	E (-V,R,1) 2 CROS
	0120	0000	
89	0122	0002	C PUMP2
	0124	0058	
90	0126	6D15	M (-10,5)
91	0128	0002	C PUMP2
	012A	0058	
92	012C	1463	E (-H,R,1) 6 CROS
	012E	0000	
93	0130	0002	C E1
	0132	007E	
94	0134	1905	M (12,1)
95	0136	1463	E (-H,R,1) 6 CROS
	0138	0000	
96	013A	0002	C E1
	013C	007E	
97	013E	7539	M (-6,14)
98	0140	1417	E (+H,R,1) 1 CROS
	0142	0000	
99	0144	1437	E (+H,R,1) 3 CROS
	0146	0000	
100	0148	1407	E (+H,R,1) 0 CROS
	014A	0000	
101	014C	14F7	E (+H,R,1) HL
	014E	0200	
102	0150	1427	E (+H,R,1) 2 CROS
	0152	0000	
103	0154	7911	M (-4,4)
104	0156	0002	C TANK
	0158	0002	
105	015A	0BF9	M (5,-2)
106	015C	100F	T (+H,B,1) L-LH
	015E	2020	
	0160	C800	
107	0162	0301	M (1,0)
108	0164	100F	T (+H,B,1) L-LM
	0166	2020	
	0168	CD00	
109	016A	0301	M (1,0)
110	016C	100F	T (+H,B,1) L-LS
	016E	2020	
	0170	D300	
111	0172	4DA1	M (-26,-24)
112	0174	100F	T (+H,B,1) N6 PILIS PUMP STATION
	0176	4E36	

	0178	2020	
	017A	5049	
	017C	4C49	
	017E	5320	
	0180	5055	
	0182	4D50	
	0184	2053	
	0186	5441	
	0188	5449	
	018A	4FCE	
113	018C	6369	M (-15,26)
114	018E	100F	T (+H,B,1) 02
	0190	30B2	
115	0192	05ED	M (2,-5)
116	0194	100F	T (+H,B,1) 5
	0196	B500	
117	0198	0701	M (3,0)
118	019A	100F	T (+H,B,1) 6
	019C	B600	
119	019E	5DF5	M (-18,-3)
120	01A0	100F	T (+H,B,1) A
	01A2	C100	
121	01A4	07F9	M (3,-2)
122	01A6	100F	T (+H,B,1) 7
	01A8	B700	
123	01AA	0BE9	M (5,-6)
124	01AC	100F	T (+H,B,1) 12
	01AE	31B2	
125	01B0	0D01	M (6,0)
126	01B2	100F	T (+H,B,1) 13
	01B4	31B3	
127	01B6	6FF9	M (-9,-2)
128	01B8	100F	T (+H,B,1) 1
	01BA	B100	
129	01BC	0701	M (3,0)
130	01BE	100F	T (+H,B,1) 2
	01C0	B200	
131	01C2	0701	M (3,0)
132	01C4	100F	T (+H,B,1) 3
	01C6	B300	
133	01C8	0701	M (3,0)
134	01CA	100F	T (+H,B,1) 4
	01CC	B400	
135	01CE	09ED	M (4,-5)
136	01D0	100F	T (+H,B,1) B
	01D2	C200	
137	01D4	695D	M (-12,23)
138	01D6	140F	T (+H,R,1)
	01D8	2020	
	01DA	2020	
	01DC	2020	
	01DE	2020	
	01E0	2020	
	01E2	20A0	
139	01E4	63C9	M (-15,-14)
140	01E6	120F	T (+H,G,1) ..
	01E8	20A0	
141	01EA	0D01	M (6,0)
142	01EC	120F	T (+H,G,1) ..
	01EE	20A0	
143	01F0	5105	M (-24,1)
144	01F2	1C0F	T (+H,N,1) DISCONNECT
	01F4	4449	
	01F6	5343	

	01F8	4F4E	
	01FA	4E45	
	01FC	43D4	
145	01FE	6FFD	M (-9,-1)
146	0200	1C0F	T (+H,N,1) PUMP
	0202	5055	
	0204	4D50	
	0206	2020	
	0208	20A0	
147	020A	6FFD	M (-9,-1)
148	020C	1C0F	T (+H,N,1) CLOSE V
	020E	434C	
	0210	4F53	
	0212	4520	
	0214	56A0	
149	0216	71FD	M (-8,-1)
150	0218	1C0F	T (+H,N,1) OPEN V
	021A	4F50	
	021C	454E	
	021E	2056	
	0220	A000	
151	0222	75FD	M (-6,-1)
152	0224	1C0F	T (+H,N,1) OR V
	0226	4F52	
	0228	2056	
	022A	A000	
153	022C	79F1	M (-4,-4)
154	022E	1C0F	T (+H,N,1) ALARM
	0230	414C	
	0232	4152	
	0234	CD00	
155	0236	8100	EOP

Appendix 2

Example of a Pump Station

Updated Program

1	0002	S TANK
2	0002 01FD	M (0,-1)
3	0004 100D	T (+V,B,1) T
	0006 D400	
4	0008 24F7	E (+H,R,2) HL
	000A 0200	
5	000C 1421	E (-V,R,1) 2 ROMB
	000E 0100	
6	0010 24F1	E (-V,R,2) VL
	0012 0300	
7	0014 1433	E (-H,R,1) 3 ROMB
	0016 0100	
8	0018 2453	E (-H,R,2) 5 CROS
	001A 0000	
9	001C 1405	E (+V,R,1) 0 ROMB
	001E 0100	
10	0020 24F5	E (+V,R,2) VL
	0022 0300	
11	0024 1417	E (+H,R,1) 1 ROMB
	0026 0100	
12	0028 0100	EOS
13	002A	S HV
14	002A 34F3	E (-H,R,3) HL
	002C 0200	
15	002E 14F3	E (-H,R,1) DIOD3
	0030 0700	
16	0032 14F3	E (-H,R,1) DIOD1
	0034 0500	
17	0036 34F3	E (-H,R,3) HL
	0038 0200	
18	003A 07FD	M (3,-1)
19	003C 100F	T (+H,B,1) V
	003E D600	
20	0040 0100	EOS
21	0042	S VV
22	0042 14F5	E (+V,R,1) VL
	0044 0300	
23	0046 14F5	E (+V,R,1) DIOD4
	0048 0800	
24	004A 14F5	E (+V,R,1) DIOD2
	004C 0600	
25	004E 14F7	E (+H,R,1) VL
	0050 0300	
26	0052 100F	T (+H,B,1) V
	0054 D600	
27	0056 0100	EOS
28	0058	S PUMP2
29	0058 0002	C TANK
	005A 0002	
30	005C 1471	E (-V,R,1) 7 CROS
	005E 0000	
31	0060 1009	T (-V,B,1) P
	0062 D000	
32	0064 01FD	M (0,-1)
33	0066 14F7	E (+H,R,1) HL
	0068 0200	
34	006A 1427	E (+H,R,1) 2 CROS
	006C 0000	
35	006E 14F1	E (-V,R,1) VL
	0070 0300	
36	0072 1463	E (-H,R,1) 6 CROS
	0074 0000	
37	0076 1403	E (-H,R,1) 0 CROS
	0078 0000	

38	007A	05FD	M (2,-1)
39	007C	0100	EOS
40	007E		S E1
41	007E	34F3	E (-H,R,3) HL
	0080	0200	
42	0082	1411	E (-V,R,1) 1 CROS
	0084	0000	
43	0086	0100	EOS
44	0088	3A0A	P 29,5
45	008A	12F3	E (-H,G,1) ROMB
	008C	0100	
46	008E	72F3	E (-H,G,7) HL
	0090	0200	
47	0092	74F3	E (-H,R,7) HL
	0094	0200	
48	0096	1405	E (+V,R,1) 0 CROS
	0098	0000	
49	009A	14F5	E (+V,R,1) VL
	009C	0300	
50	009E	1901	M (12,0)
51	00A0	1433	E (-H,R,1) 3 CROS
	00A2	0000	
52	00A4	74F3	E (-H,R,7) HL
	00A6	0200	
53	00A8	84F3	E (-H,R,8) HL
	00AA	0200	
54	00AC	1405	E (+V,R,1) 0 CROS
	00AE	0000	
55	00B0	0002	C VV
	00B2	0042	
56	00B4	05F5	M (2,-3)
57	00B6	0002	C VV
	00B8	0042	
58	00BA	05F5	M (2,-3)
59	00BC	0002	C VV
	00BE	0042	
60	00C0	05F5	M (2,-3)
61	00C2	0002	C VV
	00C4	0042	
62	00C6	75F1	M (-6,-4)
63	00C8	1477	E (+H,R,1) 7 CROS
	00CA	0000	
64	00CC	07FD	M (3,-1)
65	00CE	12F1	E (-V,G,1) VL
	00D0	0300	
66	00D2	1277	E (+H,G,1) 7 CROS
	00D4	0000	
67	00D6	0700	M (3,3)
68	00D8	54F5	E (+V,R,5) VL
	00DA	0300	
69	00DC	1EF5	E (+V,Y,1) DI0D4
	00DE	0800	
70	00F0	64F5	E (+V,R,6) VL
	00E2	0300	
71	00E4	1423	E (-H,R,1) 2 CROS
	00E6	0000	
72	00E8	A4F3	E (-H,R,10) HL
	00EA	0200	
73	00EC	54F3	E (-H,R,5) HL
	00EE	0200	
74	00F0	1EF3	E (-H,Y,1) DI0D1
	00F2	0500	
75	00F4	0002	C HV
	00F6	002A	

76	00F8	7905	M (-4,1)
77	00FA	14F5	E (+V,R,1) ROMB
	00FC	0100	
78	00FE	1701	M (11,0)
79	0100	0002	C VV
	0102	0042	
80	0104	05F5	M (2,-3)
81	0106	0002	C VV
	0108	0042	
82	010A	7DF1	M (-2,-4)
83	010C	1471	E (-V,R,1) 7 CROS
	010E	0000	
84	0110	7901	M (-4,0)
85	0112	12F1	E (-V,G,1) VL
	0114	0300	
86	0116	1247	E (+H,G,1) 4 CROS
	0118	0000	
87	011A	72F7	E (+H,G,7) HL
	011C	0200	
88	011E	1221	E (-V,G,1) 2 CROS
	0120	0000	
89	0122	0002	C PUMP2
	0124	0058	
90	0126	6D15	M (-10,5)
91	0128	0002	C PUMP2
	012A	0058	
92	012C	1463	E (-H,R,1) 6 CROS
	012E	0000	
93	0130	0002	C E1
	0132	007E	
94	0134	1905	M (12,1)
95	0136	1263	E (-H,G,1) 6 CROS
	0138	0000	
96	013A	0002	C E1
	013C	007E	
97	013E	7539	M (-6,14)
98	0140	1217	E (+H,G,1) 1 CROS
	0142	0000	
99	0144	1237	E (+H,G,1) 3 CROS
	0146	0000	
100	0148	1407	E (+H,R,1) 0 CROS
	014A	0000	
101	014C	14F7	E (+H,R,1) HL
	014E	0200	
102	0150	1427	E (+H,R,1) 2 CROS
	0152	0000	
103	0154	7911	M (-4,4)
104	0156	0002	C TANK
	0158	0002	
105	015A	0BF9	M (5,-2)
106	015C	100F	T (+H,B,1) 40H
	015E	3130	
	0160	C800	
107	0162	0301	M (1,0)
108	0164	100F	T (+H,B,1) 12M
	0166	3132	
	0168	CD00	
109	016A	0301	M (1,0)
110	016C	100F	T (+H,B,1) 06S
	016E	3036	
	0170	D300	
111	0172	4DA1	M (-26,-24)
112	0174	100F	T (+H,B,1) N6 PILIS PUMP STATION
	0176	4E36	

	0178	2020	
	017A	5049	
	017C	4C49	
	017E	5320	
	0180	5055	
	0182	4050	
	0184	2053	
	0186	5441	
	0188	5449	
	018A	4FCE	
113	018C	6369	M (-15,26)
114	018E	100F	T (+H,B,1) 02
	0190	3082	
115	0192	05ED	M (2,-5)
116	0194	100F	T (+H,B,1) 5
	0196	B500	
117	0198	0701	M (3,0)
118	019A	100F	T (+H,B,1) 6
	019C	B600	
119	019E	5DF5	M (-18,-3)
120	01A0	100F	T (+H,B,1) A
	01A2	C100	
121	01A4	07F9	M (3,-2)
122	01A6	100F	T (+H,B,1) 7
	01A8	B700	
123	01AA	0BE9	M (5,-6)
124	01AC	100F	T (+H,B,1) 12
	01AE	31B2	
125	01B0	0D01	M (6,0)
126	01B2	100F	T (+H,B,1) 13
	01B4	31B3	
127	01B6	6FF9	M (-9,-2)
128	01B8	100F	T (+H,B,1) 1
	01BA	B100	
129	01BC	0701	M (3,0)
130	01BE	100F	T (+H,B,1) 2
	01C0	B200	
131	01C2	0701	M (3,0)
132	01C4	100F	T (+H,B,1) 3
	01C6	B300	
133	01C8	0701	M (3,0)
134	01CA	100F	T (+H,B,1) 4
	01CC	B400	
135	01CE	09ED	M (4,-5)
136	01D0	100F	T (+H,B,1) B
	01D2	C200	
137	01D4	695D	M (-12,23)
138	01D6	140F	T (+H,R,1) JUNE 30 1976
	01D8	4A55	
	01DA	4E45	
	01DC	2033	
	01DE	3020	
	01E0	3139	
	01E2	37B6	
139	01E4	63C9	M (-15,-14)
140	01E6	120F	T (+H,G,1) 11
	01E8	20A0	
141	01EA	0D01	M (6,0) /
142	01EC	120F	T (+H,G,1) 52
	01EE	35R2	
143	01F0	5105	M (-24,1)
144	01F2	1C0F	T (+H,N,1) DISCONNECT
	01F4	4449	
	01F6	5343	

	01F8	4F4E	
	01FA	4E45	
	01FC	4304	
145	01FE	6FFD	M (-9,-1)
146	0200	1C0F	T (+H,N,1) PUMP
	0202	5055	
	0204	4D50	
	0206	2020	
	0208	20A0	
147	020A	6FFD	M (-9,-1)
148	020C	1C0F	T (+H,N,1) CLOSE V
	020E	434C	
	0210	4F53	
	0212	4520	
	0214	56A0	
149	0216	71FD	M (-8,-1)
150	0218	1C0F	T (+H,N,1) OPEN V
	021A	4F50	
	021C	454E	
	021E	2056	
	0220	A000	
151	0222	75FD	M (-6,-1)
152	0224	1C0F	T (+H,N,1) OR V
	0226	4F52	
	0228	2056	
	022A	A000	
153	022C	79F1	M (-4,-4)
154	022E	1C0F	T (+H,N,1) ALARM
	0230	414C	
	0232	4152	
	0234	CD00	
155	0236	8100	EOP

